DATA INSERTION AND SCRAMBLING FOR UNIFIED SCALABLE INFORMATION HIDING

ONG SIM YING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

DATA INSERTION AND SCRAMBLING FOR UNIFIED SCALABLE INFORMATION HIDING

ONG SIM YING

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Ong Sim Ying** (I.C./Passport No.: **840803-10-5626**)

Registration/Matrix No.: **WHA100032**

Name of Degree: **Doctor of Philosophy**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Data Insertion and Scrambling for Unified Scalable Information Hiding

Field of Study: **Information Hiding**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

ABSTRACT

Information hiding is generally divided into two disciplines, namely, external data insertion and scrambling. External data insertion embeds data into the multimedia content for various purposes, including enrichment (i.e., metadata and hyperlink), owner identification (i.e., fingerprint), proof of ownership (i.e., watermark), covert communication (i.e., steganography), etc., depending on the applications in question. On the other hand, scrambling conceals the perceptual meaning of the content, either completely or partially, depending on the requirement of the intended application.

Conventionally, these two disciplines are studied independently. The basic properties of these two disciplines include reversibility, embedding capacity and commutative. However, due to the massive use of digital contents in our daily life, unification of these two disciplines (hereinafter referred to as unified information hiding) is needed to cope with multiple requirements in a single application.

In this thesis, the background of information hiding, research motivations, objectives and methodology are presented in **Chapter 1**. Next, **Chapter 2** proposes the classification of unified information hiding and reviews the relevant state-of-the-art methods. The following chapters propose four different unified information hiding approaches in the spatial and compressed domains for image.

Specifically, in **Chapter 3**, an Embed-To-Scramble method, namely, Histogram Association Mapping (HAM), is proposed to overcome the problems in conventional histogram manipulation method in the spatial domain. An image is first divided into non-overlapping blocks, and each block is further classified into two categories to perform their respective designated scrambling-embedding operations. This method is reversible, scalable and the embedding capacity is relatively high when compared to the conventional methods.

Chapter 4 puts forward a scalable Scramble-Then-Embed method named SURIH to realize scrambling-embedding in the Discrete Cosine Transform (DCT) domain. Adaptive scanning order is proposed to suppress the bitstream size increment caused by encoding the side information of embedding technique. Unlike the conventional methods that confine their scope of permutation to a block, SURIH permutes the coefficients globally throughout the entire image. Performance of SURIH is evaluated at the end of the chapter.

In **Chapter 5**, two Scramble-To-Embed methods are proposed in the spatial and DCT compressed domain, in which the recovery processes rely on the natural properties of the utilized components. The utilized components in the DCT compressed methods include DC coefficient, AC Block Energy (i.e., sum of magnitude of each 8×8 block), and ZRV (zero-run, AC coefficient value) pairs. Reversibility is achieved by exploiting the natural property of each component, along with the aid of minimum amount of side information. Similar techniques are also applied in the spatial domain, and it is further discussed in this chapter. At the end of the chapter, experimental results are recorded and compared with the conventional methods.

Chapter 6 discusses the important properties achieved in each proposed method and their limitations in detail. Specifically, the embedding capacity and image quality can be gradually controlled by tuning the parameters introduced in each proposed method. The reversibility and commutative property of each method are also discussed.

Finally, **Chapter 7** concludes this thesis and discusses the future work of scalable unified information hiding.

Abstrak

Secara umumnya, *information hiding* dibahagikan kepada dua disiplin, termasuk *external data insertion* dan *scrambling*. *External data insertion* menyimpan data ke dalam kandungan multimedia untuk pelbagai kegunaan, termasuk penambahan maklumat (metadata dan pautan), pengenalan pemilik (*fingerprint*), kebuktian hakmilik (tera air), komunikasi rahsia (steganografi), dan sebagainya, bergantung kepasa aplikasi yang berkenaan. Manakala *scrambling* menyembunyi maksud persepsi bagi kandungan multimedia, secara keseluruhan atau sebahagian, bergantung kepada keperluan dan tujuan aplikasi tersebut.

Lazimnya, dua disiplin ini dikaji secara berasingan. Sifat-sifat asas bagi kedua-dua disiplin ini termasuk *reversibility*, kapasiti *embedding*, dan *commutative*. Disebabkan oleh kegunaan kandungan digital yang banyak, penyatuan dengan menggunakan kedua-dua disiplin ini (selepas ini dipanggil sebagai *unified information hiding*) adalah diperlukan untuk memenuhi pelbagai kehendak di dalam satu aplikasi.

Dalam tesis ini, latar belakang bagi *information hiding*, motivasi penyelidikan, objektif dan metodologi akan dibentang di **Chapter 1**. Justeru itu, **Chapter 2** mengemukakan cadangan klasifikasi bagi *unified information hiding* dan mengulaskan kaedah-kaedah konvensional yang berkenaan. Bab-bab yang berikut mencadangkan empat kaedah *unified information hiding* di dalam domain spatial dan pemampatan bagi imej.

Kaedah *Embed-To-Scramble* yang bernama *Histogram Association Mapping* (HAM) telah dicadangkan di **Chapter 3** untuk mengatasi masalah-masalah yang dihadapi oleh kaedah *histogram manipulation* yang konvensional di dalam domain spatial. Imej akan dibahagikan kepada blok-blok yang tidak bertindih, dan selepas itu, setiap blok akan diklasifikasikan kepada dua kategori untuk menjalankan operasi *scrambling-embedding* masing-masing. HAM adalah *reversible*, *scalable*, dan kapasiti *embedding*-nya lebih tinggi berbanding dengan kaedah-kaedah konvensional yang lain.

Chapter 4 mengemukakan satu kaedah *Scramble-Then-Embed* bernama SURIH untuk merealisasikan *scrambling-embedding* di dalam domain *Discrete Cosine Transform*. *Scanning order* yang adaptif telah dicadangkan untuk menyekat saiz *bitstream* yang meningkat, diakibatkan oleh pengekodan *side information* sewaktu process *embedding*. Berbeza dengan kaedah konvensional yang mengehadkan skop permutasi di dalam satu blok, SURIH menjalankan operasi permutasi secara globa, iaitu, di dalam seluruh imej. Akhirnya, prestasi SURIH akan diperiksa dan dinilai pada penghujung bab ini.

Sementara bagi **Chapter 5**, dua kaedah *Scramble-To-Embed* telah dicadangkan dalam domain spatial dan DCT pemampatan, di mana process penyahkodan bergantung kepada ciri semularjadi bagi komponen yang digunakan. Komponen yang digunakan di domain DCT pemampatan termasuk *DC coefficient*, *AC Block Energy* (iaitu jumlah magnitud dalam setiap 8×8 blok), dan *ZRV* (*zero-run*, nilai *AC coefficient*) *pairs*. Dalam kaedah ini, *reversibility* dicapai dengan mengeksplotasi ciri-ciri semulajadi bagi setiap komponen, dengan bantuan *side information* yang minima. Teknik yang serupa juga digunakan di dalam domain spatial, dan butiran penggunaan ini akan dibincang di dalam bab ini. Pada hujung bab ini, hasik eksperimen bagi kedua-dua kaedah akan dikemukakan dan dibandingkan dengan kaedah-kaedah konvensional yang lain.

Chapter 6 membincang dengan terperinci sifat-sifat penting dan pembatasan yang dihadapi oleh setiap kaedah yang dicadangkan dalam tesis ini. Khususnya, kapasiti *embedding* dan kualiti imej yang boleh dikawal secara ansuran, dengan penyelarasan parameter-parameter yang digunakan masing-masing. Sifat *reversibility* dan *commutative* bagi setiap kaedah juga dibincang dalam bab ini.

Chapter 7 yang terakhir menyimpulkan tesis ini dan membincang kajian-kajian yang berpotensi pada masa depan dengan menggunakan *unified information hiding* yang *scalable*, seperti yang dicadangkan dalam tesis ini.

ACKNOWLEDGEMENTS

This thesis is dedicated to my beloved parents, sisters and fiancé, for their continuous and endless support. It is easy to start the Ph.D journey, but it is a challenging task to finish it. One requires not only self-persistency, open and positive mind, but also the encouragements from family and friends. Therefore, I would like to thank my family and friends for always being there for me.

My supervisor, Dr. KokSheik Wong is a passionate, caring, inspiring and supportive person. He is the one who inspired me in my research in the first place and then been through all the obstacles together with me to reach my goal. He is the friend who advised me and taught me patiently throughout my Ph.D journey. I could never complete my Ph.D without his guidance. I would like to express my greatest gratitude to Dr. KokSheik Wong for his invaluable supports and assistance.

Next, I would also like to acknowledge to my mentor, Prof. Kiyoshi Tanaka, from Shinshu University, Japan, with much appreciation. He is a kind, caring, and thoughtful person. He has enlightened me with the clear direction for research and has shared his research experiences with me in an openly manner. He is like a father to all his students. I cherish all our research discussions and enjoyable leisure moments together.

In addition, I want to say thank you to all my research mates in Multimedia Signal Processing and Information Hiding (MSPIH) laboratory, University of Malaya. We have worked together to achieve different goals, and have created a great research atmosphere. Last but not least, I would also like to thank SBUM (UM Fellowship Scheme) and University of Malaya for the financial means and laboratory facilities provided.

Obtaining a Master or Ph.D degree is of course not the end of our research journey; it is only a preparation for us to go further in the following stage of our life. I want to thank you again for everyone who journeyed with me in this important part of my life. I

truly appreciate and feel grateful for having every one of you in my life.

To better future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Contemporary technology provides the infrastructure for people to reach for information easily in the digital form. We can conveniently capture, store, administrate, alter, and remove the information using the affordable yet powerful digital devices. Everyday, we transmit digital information for various purposes. We send text messages, share pictures on social networking service website, exchange emails, converse through Internet messengers, store our data in clouds, etc. All these activities are carried out with the assumption that the processes involved in the communications are safe. Yet, the conveniences given by the technology not only bring benefits to the rightful users, but it also brings danger.

Information hiding is a common way to obscure the data in communication channel and storage. Previous researches invented numerous algorithm such as DES (46-3, 1999), AES (197, 2001), etc. to protect the highly sensitive data such as password, credit card transaction, etc (Yang, Bourbakis, & Li, 2004). Similar to the traditional safe in the bank, these data are securely locked in mathematical safe. We refer to this type of information hiding as encryption. Soon after, watermark and fingerprint are included in the commercial products such as audio, image, good, etc. to claim ownership or trace illegal distributors. This is another way to hide information, which we refer to as external data insertion.

Nowadays, among all the digital information, image is undeniably the most widely consumed content. Although there are many established formats, such as TIFF, PNG, JPEG-XR, GIF, etc., JPEG is still the mostly deployed format due to its simplicity and good trade-off between compression ratio and quality. In addition, most of the smart-

phones and camera devices are capturing in the JPEG format. Hence, in this thesis, information hiding are investigated in both the spatial and JPEG compressed domains.

1.1 Information Hiding

Information hiding is the art and science of providing secrecy to communication (Neil F. Johnson & Jajodia, 2001), in a designed manner to achieve its desired purposes. It is generally pursued in two major directions, namely, encryption and external data insertion (Rad, Wong, & Guo, 2014).

Encryption conceals the perceptual meaning of the multimedia content by making it unintelligible (i.e., resemble white noise) (Van De Ville, Philips, Van De Walle, & Lemahieu, 2004; Karim & Wong, 2014). As an alternatively to encryption, scrambling is proposed to overcome certain application and implementation constraints such as computational power, memory, and format compliance while slightly compromising robustness (Engel, Pschernig, & Uhl, 2008). Scrambling is also commonly referred to as lightweight encryption, perceptual encryption, or transparent encryption in the literature. More often than not, scrambling is achieved using permutation (i.e., change of position) and substitution (i.e., replacing with other value). By design, scrambling is suitable for the application requiring lower security and application which needs to process a large amount of data. Hereinafter, scrambling in also referred to as quality degradation.

On the other hand, for external data insertion, image content is also referred to as carrier (or cover) because it is utilized to hold external data. The term "data" here is a generalized way to represent the data embedded into the image content and it differs depending on the application in question. For example, data such as the checksum and hash value of the carrier are inserted into the carrier to verify the integrity of the carrier. Watermark such as logo, unique vendor information, etc. are often inserted into the host to claim the ownership of the image content (Lusson, Bailey, Leeney, & Curran, 2013)

or for tampering detection purposes (Battisti, Cancellaro, Boato, Carli, & Neri, 2009). Fingerprint of the buyer are embedded into the distributed videos, images, and audio to trace the illegal distributor and enforce copyright (Kundur & Karthik, 2004). Secret message between two parties can also be communicated covertly using steganographic techniques, which hides the information without rising the attention of the third party. In addition to security-related application, other interesting applications such as hyperlink and metadata insertion are utilized to enrich the image content (Tew & Wong, 2014; Mobasseri, Berger, Marcinak, & NaikRaikar, 2010) or digital archiving purposes (C.-W. Lee & Tsai, 2013).

Past researches focused on the study of these two disciplines independently. However, single purpose method is no longer sufficient to cope with the needs of the current society. Each discipline provides some amount of misdirection and some amount of security (Wayner, 2009). Thus, many of the current applications incorporate scrambling and data insertion to complement each other (X. Wu & Sun, 2014; Ma, Zhang, Zhao, Yu, & Li, 2013; X. Zhang, 2012; Kundur & Karthik, 2004). We call the integration of these two disciplines as Unified Information Hiding (UIH). More information about scrambling, external data insertion and their unification are discussed in **Chapter 2**.

1.2 Motivations and Objectives

The motivation of this study is to unify scrambling and data insertion to complement each other. This study focuses on unifying information hiding for image in both the spatial and compressed domains. In addition, a general framework is proposed to capture different approaches of unification. It is important to study how the unification methods are able to provide more flexibility by relaxing certain requirements (e.g., image quality) and the common properties that they share.

The main objectives of the thesis are:

1. To study scrambling, data insertion and unified information hiding methods with reversible and scalable functionalities.

2. To propose a framework for unified information hiding and categorize the conventional methods into the framework.

3. To propose new instances of unified information hiding method with reversible and scalable functionalities based on the general framework.

4. To evaluate the proposed unified information hiding methods.

In addition than fulfilling the main objectives, the problems encountered by each proposed method are identified and addressed in the following chapters:

5. **Chapter 3** proposes a histogram manipulation method (namely, HAM) using E2S (Embed-To-Scramble) approach, to embed data into the image while scrambling the image to achieve scalability and reversibility. In addition, the proposed HAM also addresses the following problems of the traditional histogram shifting method:

    a) Expensive preprocessing,

    b) Overflow and underflow problem,

    c) Low embedding capacity, and

    d) Unable to embed data when all bins in the histogram are occupied.

6. **Chapter 4** proposes a unified method called SURIH in JPEG compressed domain, using STE (Scramble-Then-Embed) approach to realize scrambling-embedding while achieving scalability and reversibility. Furthermore, the common problems in the conventional information hiding methods in JPEG compressed domain are also addressed:

a) Limited scope of permutation, and

b) Bitstream size increment for the output image.

7. **Chapter 5** proposes unified method in the spatial domain (namely, NPIS) and JPEG compressed domain (namely, NPIC) using S2E (Scramble-To-Embed) approach, to scramble the image while embedding data into it to achieve scalability and reversibility. Natural properties of image (NPI) in both the spatial and JPEG compressed domains are exploited in the proposed methods. In the spatial domain, the proposed NPIS method has to overcome the following problems:

   a) Eliminate blockiness and visible outline occurred in HAM in **Chapter 3**, and

   b) Enhance scope of scrambling by introducing permutation method in HAM.

   In the JPEG compressed domain, the proposed NPIC method has to overcome the following problems:

   a) Further suppress the bitstream size increment for SURIH in **Chapter 4**, and

   b) Fully utilize all the components in JPEG for scrambling-embedding.

1.3 Research Methodology

Figure 1.1 shows the research methodology employed in this study. Literature review involves the study on the current state-of-the-art methods in scrambling, data insertion and the unified methods. The advantages and disadvantages of all the considered methods are analyzed. Then, the research objectives of this research are defined. A general framework of unified information hiding is proposed, and the conventional methods reviewed are fitted infer the framework for classification purposes. In this study, we proposed unified information hiding methods in both the spatial and frequency domains using different techniques. Thus, specific objectives are defined in each chapter. Then,

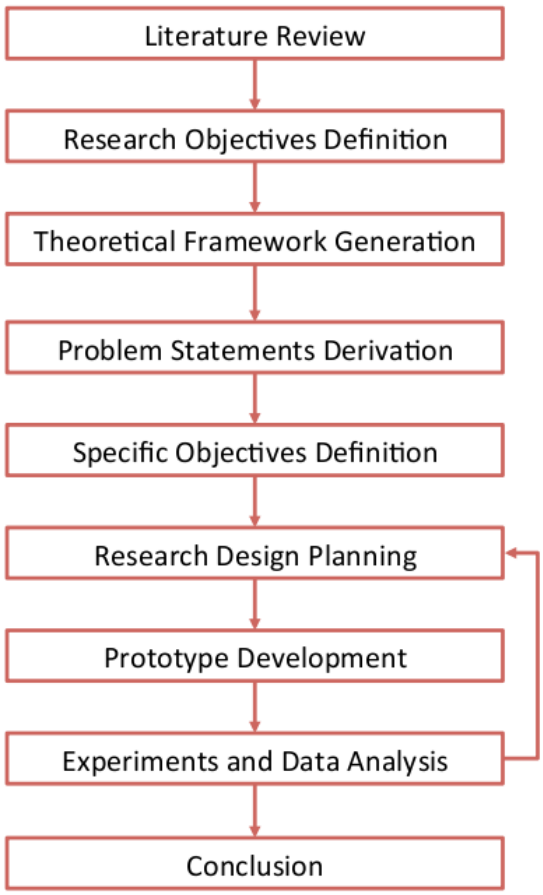


Figure 1.1: Research Methodology

the research design is drafted based on the general and specific objectives to be achieved in every domain. Based on the research design, prototype is built and tested. Results are then gathered and analyzed to investigate the performance of each method. The research design is re-adapted based on the analysis. At the end, conclusion are drawn.

1.4 Thesis Outline

This thesis consists of seven chapters. This chapter gives a brief introduction on information hiding, defines the objectives of the thesis and describe the research methodology employed throughout the study. The next chapter discusses and analyzes the frameworks for unified information hiding and their related works. **Chapter 3** proposes a unified information hiding method in the spatial domain using the E2S approach while **Chapter 4** proposes a unified method in the compressed domain using the STE approach. **Chapter 5** proposes another unified method in the spatial and compressed domains using

the S2E approach to solve the problems in **Chapter 3** and **Chapter 4**. Next, **Chapter 6** reviews the important properties achieved in each proposed method and identify their limitations. Finally, **Chapter 7** discusses the future work and outcome of the thesis, and lastly, concludes the thesis.

CHAPTER 2

UNIFIED INFORMATION HIDING

Over the years, information hiding has been persued in two general directions, namely scrambling and external data insertion (Katzenbeisser & Petitcolas, 2000). These disciplines play important roles in digital content management and protection. Recently, they are gaining more attention due to the vast availability of capturing devices at affordable prices and the ever improving internet facilities to share contents in more convenient ways. Among the multimedia contents, image is arguably the mostly considered one. It is reported that 240,000 photos are uploaded each minute to Facebook (*IACP, Center for Social Media*, 2014).

Although scrambling and data insertion differ in many aspects, they are often utilized to complement each other (Battisti et al., 2009). We called this integration as Unified Information Hiding (UIH). UIH integrates data insertion and scrambling to provide diverse functionalities to a single application. In this chapter, the framework of unified information hiding is proposed and the related works in UIH are reviewed. In particular, **Section 2.1** and **Section 2.2** reviews the related methods for external data insertion and scrambling, and their encoding/decoding processes, respectively. **Section 2.3** puts forward the framework to capture different approaches of UIH and analyzes their associated methods. Subsequently, **Section 2.4** presents the properties of UIH and **Section 2.5** states some of the possible applications using UIH. Lastly, **Section 2.6** summarizes this chapter.

2.1 External Data Insertion (Embedding)

External data insertion embeds data into the image. The external data inserted can be of any form, depending on the application in question. For example, watermark

for ownership identification (Lusson et al., 2013; De Vleeschouwer, Delaigle, & Macq, 2003; Tian, 2002; Fridrich, Goljan, & Du, 2001, 1900), fingerprint for copyright protection (Kundur & Karthik, 2004), secret message for steganography (Luo, Huang, & Huang, 2010; Sur, Goel, & Mukhopadhyay, 2008; X. Zhang & Wang, 2005; Kawaguchi & Eason, 1998), metadata for multimedia enrichment (Mobasseri et al., 2010), etc. Each of these applications has their very own unique requirements (i.e., properties) to achieve their objective. Nevertheless, the commonly considered properties among all data insertion methods include reversibility, high embedding capacity and high output image quality. Specifically, reversibility in external data insertion is the ability to fully recover the carrier after data extraction process. Embedding capacity is the number of embeddable bits (or sometimes measured in bits per pixel) into the carrier, and output image quality is the quality of the embedded image (i.e., outcome of data insertion process).

The data insertion process embeds data into the input image (also known as carrier or host image) with or without the secret key to produce the embedded image (see **Figure 2.1**). On the other hand, the data extraction process extracts the embedded data from the embedded image with the correct secret key (if needed) and produce the output image (see **Figure 2.2**).

The earliest way to perform data insertion in digital image is by using Least Significant Bit (LSB) insertion (Hong, 2012; Luo et al., 2010; Celik, Sharma, Tekalp, & Saber, 2005; Chan & Cheng, 2004). The LSB of the pixels (Chan & Cheng, 2004) or coefficients (Upham, 1999) are modified based on the external data. Note that, in this thesis, all the external data mentioned is presented in the binary form. Therefore, if the external data is '0', the encoder (i.e., encoding algorithm) changes the LSB of the selected pixel to 0. Otherwise, it will change the LSB to 1 to represent the external data of '1'. This approach is simple and able to retain high quality in the output image. Nevertheless, the major drawback of this method is that the external data can be easily detected (Thai, Cogranne,

& Retraint, 2014), and modified by the attacker. Another widely considered approach is to modify the LSB by means of adding or subtracting unity to match the external data. To improve image quality, Luo et al. proposed an enhancement of LSB Matching Revisited method by integrating LSB embedding with an edge-adaptive algorithm (Luo et al., 2010). More sophisticated methods such as multiple-based notational data embedding method (X. Zhang & Wang, 2005) and Bit-plane complexity segmentation (Kawaguchi & Eason, 1998) are proposed to exploit the texture of the host image.

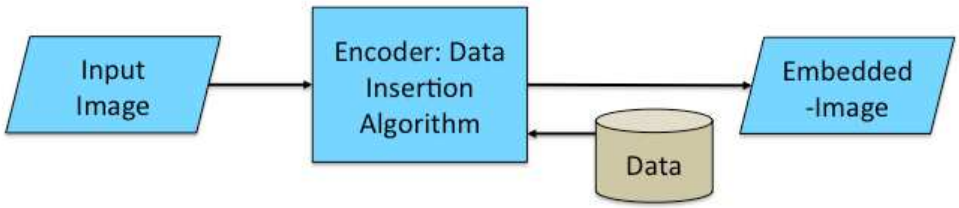


Figure 2.1: Encoding - Data Insertion Process

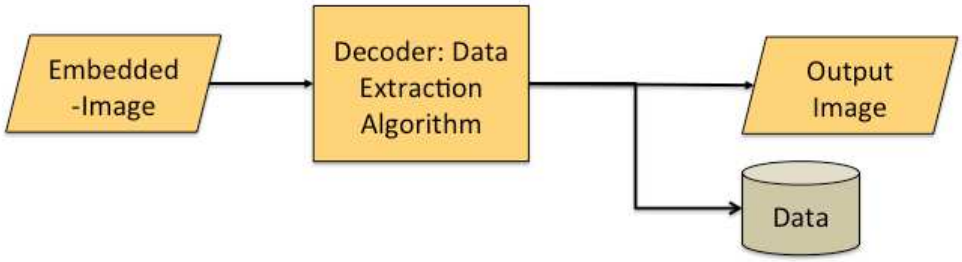


Figure 2.2: Decoding - Data Extraction Process

For reversible method, Vleeschouwer et al. proposed to embed data into the image by changing the center of mass in the histogram of a block of pixels (De Vleeschouwer et al., 2003). Although this method is reversible, its embedding capacity is low. Another popular way to embed data is by using reversible histogram shifting proposed by Ni et al. (Ni, Shi, Ansari, & Su, 2006). Histogram of the image (i.e., x-axis is the pixel value while y-axis is the number of occurrences for each value) is utilized for data insertion purposes. First, histogram shifting method selects the bin to embed data (i.e., select the bin of highest number of occurrences to maximize the capacity). Next, all bins to the left

of the selected bin are shifted to the left (i.e., minus one) to vacate an empty bin for data embedding purposes. Then, by using the combination of selected bin and empty bin, one bit can be embedded for every occurrence of the selected pixel value. For example, one bin can be used to encode '0' while emptied bin can be used to encode '1'. Histogram shifting maintains the image quality at a high level, but the preprocessing (i.e., shifting) is significant. There are many different variations and improvements of histogram shifting method (X. Li, Li, Yang, & Zeng, 2013; Huang & Fang, 2011) and they will be further discussed in **Chapter 3**.

Difference expansion (DE) method (Dragoi & Coltuc, 2014; Hong, 2012; Tian, 2002) is also commonly utilized for data insertion purposes. Pixels are grouped in pairs and the differences between pairs are calculated. Errors (i.e., differences) are expended (e.g., doubled) to create the space for data embedding. Then the LSB of the multiplied error is modified based on the external data to be embedded. This method is reversible. Data is embedded using the errors, which reduces the distortion of the modified pixels and therefore the output image is of high quality. However, the embedding capacity is relatively low. Similarly, Hong et al.'s method realized data insertion using DE, but this method is implemented using various block sizes based on quadtree partitioning. The complexity of the image determines the block sizes and amount of external data to be embedded (Hong, 2012). Hence, Hong et al.'s method (Hong, 2012) achieves higher capacity and lower distortion when compared to Tian et al.'s method (Tian, 2002). Thodi et al. then proposed to embed data in the prediction errors (C.-F. Lee & Chen, 2012; Thodi & Rodriguez, 2004) to improve the performance of DE method by reducing the distortion in image quality and increasing the embedding capacity.

The aforementioned general methods can be implemented in all formats, such as pixel, coefficients, etc. In particular, format-compliant methods are those that can only be utilized by using their specific components. For example, data embedding method us-

ing quantizer (C.-C. Lin, Chen, & Hsueh, 2009) in JPEG (Joint Picture Expert Group) and MPEG (Motion Picture Expert Group), methods using motion vector in H.264/AVC (Z. Liu, Liang, Niu, & Yang, 2004; Bodo, Laurent, & Dugelay, 2004; J. Zhang, Li, & Zhang, 2001), method using Huffman table modification (Xu, Wang, & Shi, 2014; Kankanhalli & Guan, 2002), etc.

Nevertheless, regardless of the method and application in question, capacity, reversibility, and output image quality are always considered to find a good trade-off among them.

2.2 Scrambling

From the perspective of content (image) management, encryption converts meaningful information into incomprehensible form that appears completely random (Wayner, 2009). One of the common use of encryption is to secure the transmission via public communication channels (i.e., the Internet) to prevent unauthorized viewing, illegal redistribution, etc. However, encryption often generates output that resembles noise, which complicates the operationality of the succeeding process such as external data insertion and compression (Wayner, 2009). Therefore, partial encryption (hereinafter referred to as scrambling) is introduced to conceal the perceptual meaning of the content, fully, or partially. In this study, we only focus on scrambling the entire image, although partial scrambling hides the region-of-interest or important region in the image (Grangetto, Magli, & Olmo, 2006). Properties involved in scrambling includes reversibility and scrambling strength. Reversibility is important in scrambling because a good scrambling algorithm recovers the scrambled image to its original state. In addition, scrambling strength measures the number of trials needed by the attacker needed to break the encoding algorithm and restore the scrambled image.

Figure 2.3 and **Figure 2.4** show the encoding and decoding processes for scram-

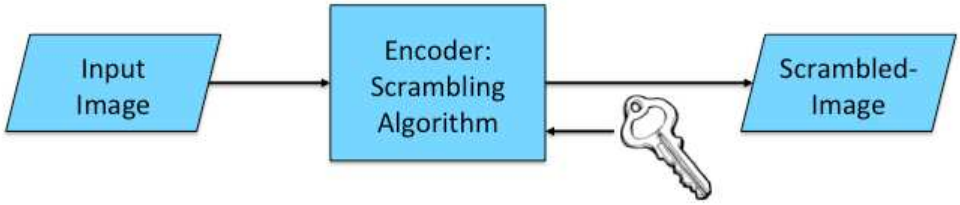


Figure 2.3: Encoding - Scrambling Process

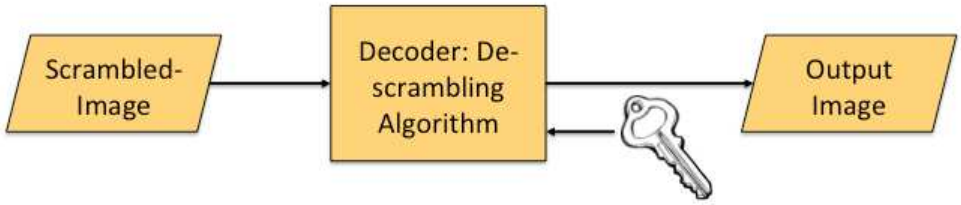


Figure 2.4: Decoding - Descrambling Process

bling. Scrambling process permutes and/or substitutes the image content while de-scrambling process recovers the scrambled image to its original state.

One of the most common ways to perform scrambling is by using the XOR operation (Hu & Han, 2009). Hu et al. propose row-based and column-based scrambling methods to increase its scrambling strength. A key is utilized to select the row(s) or column(s) to be the key-row or key-column (Hu & Han, 2009). Other rows or columns are XOR-ed by using the key-row or key-column to produce the scrambled image. Wong et al. propose another scrambling method called Unified Constructive Permutation Function (UCPF) by using permutation technique (Wong & Tanaka, 2010a). This technique shuffles the location of the pixels alternately by using a key. Then, a noise function is added to increase the scrambling strength by further distorting the shuffled-image. Besides, Wu et al. proposed to generate a sudoku matrix which will be utilized to modify the pixel values and locations in the image to achieve distortion (Y. Wu, Zhou, Noonan, Panetta, & Agaian, 2010). These aforementioned methods are simple and effective, but they cannot be applied directly to compressed domain.

In compressed domain (e.g., in DCT domain), the scrambling methods often fo-

cuses on Direct Current (DC) and Alternating Current (AC) coefficients. Slight modification such as changing the location of coefficients might cause large bitstream increment. Therefore, the trade-off between the scrambling strength and bitstream size increment must be judiciously maintained. A simple way to achieve scrambling in compressed domain is by flipping the sign of nonzero coefficients (Kundur & Karthik, 2004; Shi & Bhargava, 1998). However, this method is not reversible. Tang splits the DC coefficients to achieve scrambling in DCT domain (Tang, 1996). By modifying the DC coefficient, the perceptual meaning of the compressed image is concealed because DC coefficients contains the overall energy of a coefficient block. Nevertheless, adding an additional non-zero AC coefficient in every block increases the bitstream size of the compressed image. On the other hand, AC coefficients are modified to achieve scrambling, including intra-block shuffling (Tang, 1996) and inter-block shuffling (X. Liu & Eskicioglu, 2003). In particular, intra-block shuffling randomizes the location of 63 AC coefficients in a block while inter-block shuffling permutes the location of the block throughout the image.

However, Li et al. propose the non-zero coefficient count attack (NZCA) (W. Li & Yuan, 2007) to test the robustness of the scrambling methods. In particular, (X. Liu & Eskicioglu, 2003; Shi & Bhargava, 1998; Tang, 1996) failed to survive NZCA. Li et al. then propose Full Inter-Block Shuffling (FIBS) which shuffles the coefficients within the same frequency sub-band. However, FIBS breaks the original characteristic exploited by zigzag scanning order, which leads to the increment of the bitstream size of the output image. Analysis of the related works in DCT domains are further discussed in **Chapter 5**.

2.3 Framework for Unified Information Hiding

Figure 2.5 and **Figure 2.6** illustrate the UIH encoding and decoding frameworks and each consists of 4 components. In the encoding framework, the top-left component is

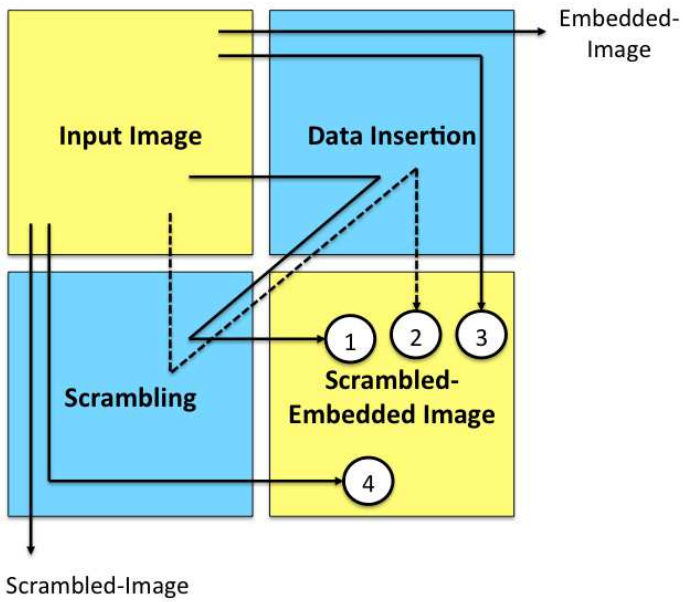


Figure 2.5: Encoding Framework for Unified Information Hiding

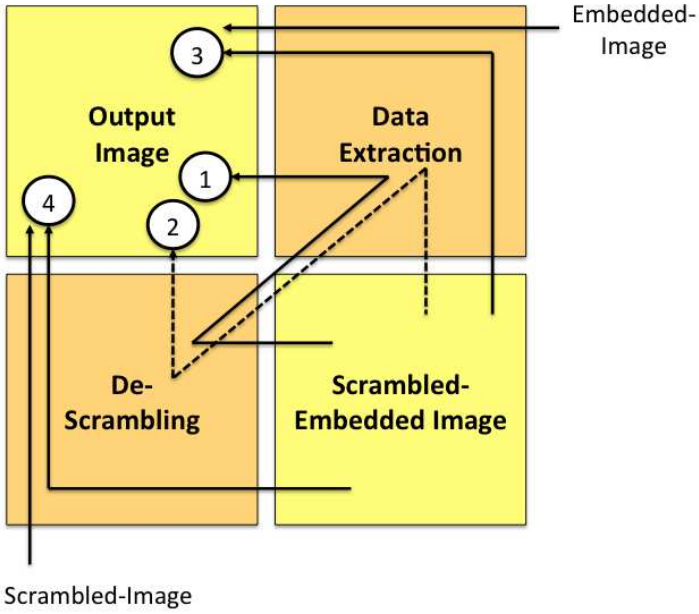


Figure 2.6: Decoding Framework for Unified Information Hiding

the input (i.e., image), while the bottom-right component is the scrambled and embedded image (i.e., output), which is the outcome of all unified information hiding method. Then, the top-right and bottom-left components are intermediate (viz., transit) states involved in unified information hiding, namely, the scrambling and data insertion states, respectively. Based on **Figure 2.5**, there are four possible approaches to achieve unified information hiding, namely: (1) Embed-Then-Scramble (ETS), (2) Scramble-Then-Embed (STE), (3)

Embed-To-Scramble (E2S), and (4) Scramble-To-Embed (S2E). The encoding and decoding processes for each approach is detailed in the rest of this subsection.

The first approach (i.e., ETS) achieves unified information hiding by using the combinations of two techniques (Ma et al., 2013; Memon & Wong, 2001). As shown in **Figure 2.5** (see path 1), data is inserted into the image content (i.e., carrier), then scrambling is performed. Ma et al. proposes Reserving Room Before Embedding (RRBE) method to realize unified information hiding, and it is categorized under the ETS approach. Pixels are first divided into reserved pixels and embedding pixels (Ma et al., 2013). The LSB of the embedding pixels are inserted into the LSB of the reserved pixel using a reversible data insertion method. Next, the external data is embedded into the LSB of embedding pixels. The output image is then scrambled to produce a scrambled-embedded image. This method is simpler than other ETS UIH methods but the capacity is restricted by the underlying reversible data embedding method in use. Another possible application is to fingerprint a content (to trace pirate involved in illegal redistribution) and then scramble the content prior to transmission to avoid unauthorized viewing. This fingerprinting application is then evolved into the buyer-seller watermarking protocol (Memon & Wong, 2001). All in all, ETS is the easiest approach to achieve unified information hiding. However, the drawback of this unification is that the sequence of the decoding is restricted (see path 1 in **Figure 2.6**).

Similar to ETS, Scramble-Then-Embed (STE) utilizes two techniques to achieve unified information hiding (see path 2 in **Figure 2.5**). However, the image content is first scrambled then data is inserted into the scrambled-image. It is sometimes referred to as data embedding in encrypted domain. Conventional data embedding algorithms often exploit the correlation or redundancy of the carrier to perform data insertion. However, the correlation and redundancies in the scrambled-domain are demolished by the scrambling operations. Therefore, it is somehow more technically challenging to design a UIH

method following in ETS approach. Zhang et al. proposes to embed data into the LSB of the encrypted pixels (X. Zhang, 2012). LSB is an exception method which does not embed data relying on the correlation or redundancy. However, Zhang et al.'s method cannot guarantee the reconstruction of the image and the embedding capacity is very low. Since the data is embedded in LSB, it can be removed and retrieved easily, which is the same problem suffered by the conventional LSB insertion method. Recently, Fujiyoshi proposed a separable unified method by encrypting and vacating room for data embedding using histogram permutation (Fujiyoshi, Kuroiwa, & Kiya, 2008). Although this method is reversible, the embedding capacity is limited and it is designed to operate in the spatial domain. In summary, similar to ETS approach, the order of decoding in any STE approach is also restricted (see arrow number 2 in **Figure 2.6**). In this thesis, **Chapter 4** proposes an unified method using STE approach in the JPEG compressed domain.

On the other hand, Embed-To-Scramble (E2S) and Scramble-To-Embed (S2E) are the novel unified approaches which utilize only a single technique (i.e., either scrambling or embedding technique) to produce scrambled-embedded output. E2S redesigns data insertion technique to achieve data embedding while scrambling the carrier (see path 3 in **Figure 2.5** and **Figure 2.6**). The parameters in the data embedding method can be exploited to control the distortion and embedding capacity as detailed in **Chapter 3**.

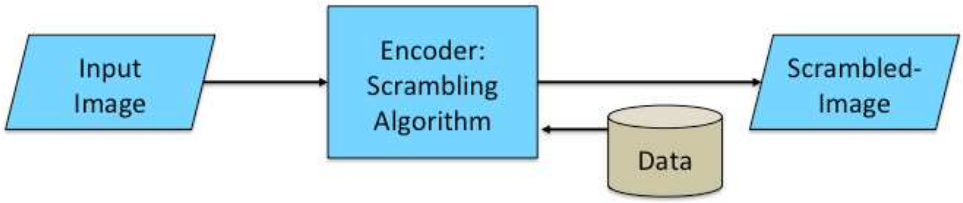


Figure 2.7: Encoding - S2E Approach

On the another hand, S2E redesigns scrambling technique to scramble the carrier while embedding data into it (see path 4 in **Figure 2.5**). The S2E approach can be realized by replacing the key in **Figure 2.3** with external data as shown in **Figure 2.7**. Con-

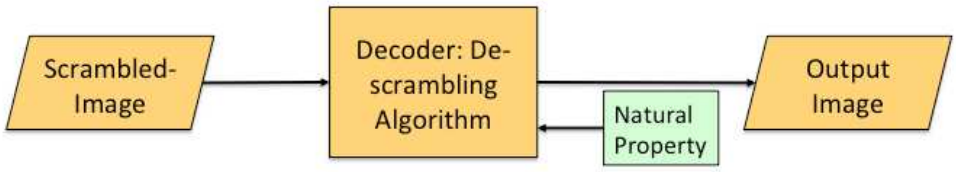


Figure 2.8: Decoding - S2E Approach

ventional scrambling method utilizes a key as the seed to decide the random sequence of scrambling for the output image. However, in the S2E approach, external data is utilized to decide the sequence of scrambling (i.e., arrangement) for producing the scrambled-embedded image. The biggest challenge in this approach (i.e., **Figure 2.7**) is how to perform the decoding process (i.e., image reconstruction and external data extraction) without the key. As a matter of fact, conventional scrambling method utilizes the same key for scrambling in the descrambling process to recover the image. However, there is no key in this approach. Hence, natural properties in the image are exploited (as shown in **Figure 2.8**) in the S2E approach to perform the reversing process. This thesis exploits the natural properties of image in the spatial and compressed domains and they are further discussed in **Chapter 5**.

2.4 Properties in UIH

2.4.1 Reversibility

Reversibility is a common property for scrambling and data insertion. A method is called reversible in UIH if the scrambled-embedded image can be fully recovered to its original state after undergoing the descrambling and data extraction processes as shown in **Figure 2.9**. Otherwise, it is called irreversible method due to the permanent changes made during scrambling or embedding (Goljan, Fridrich, & Du, 2001). Reversibility is a crucial requirement for applications such as medical, military, forensic, historical artwork preservation, etc., where any form of distortion is not permissible (De Vleeschouwer et al., 2003).

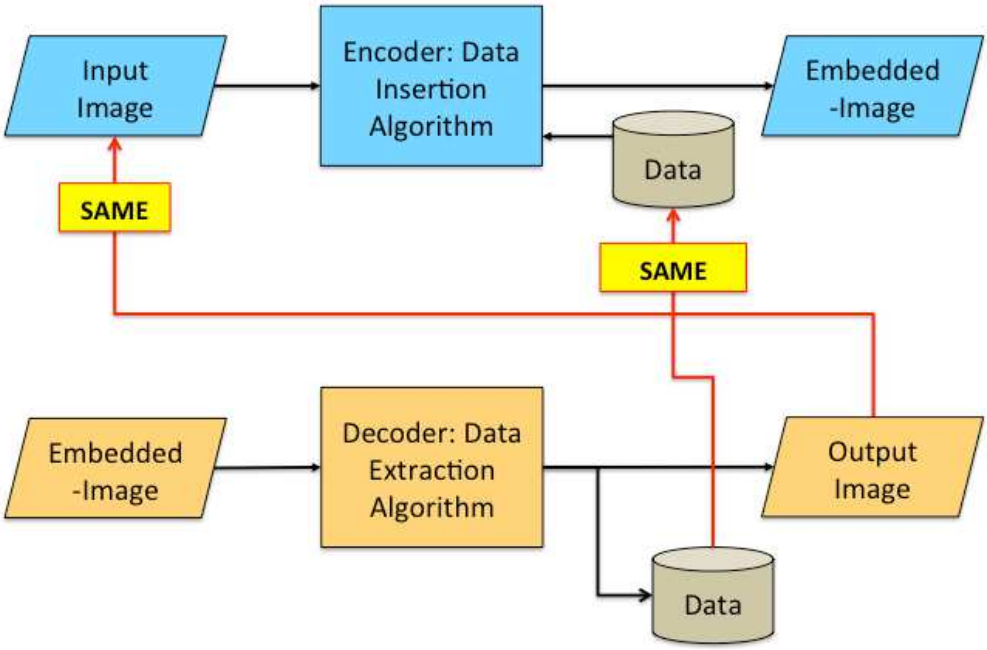


Figure 2.9: Reversibility in Data Insertion

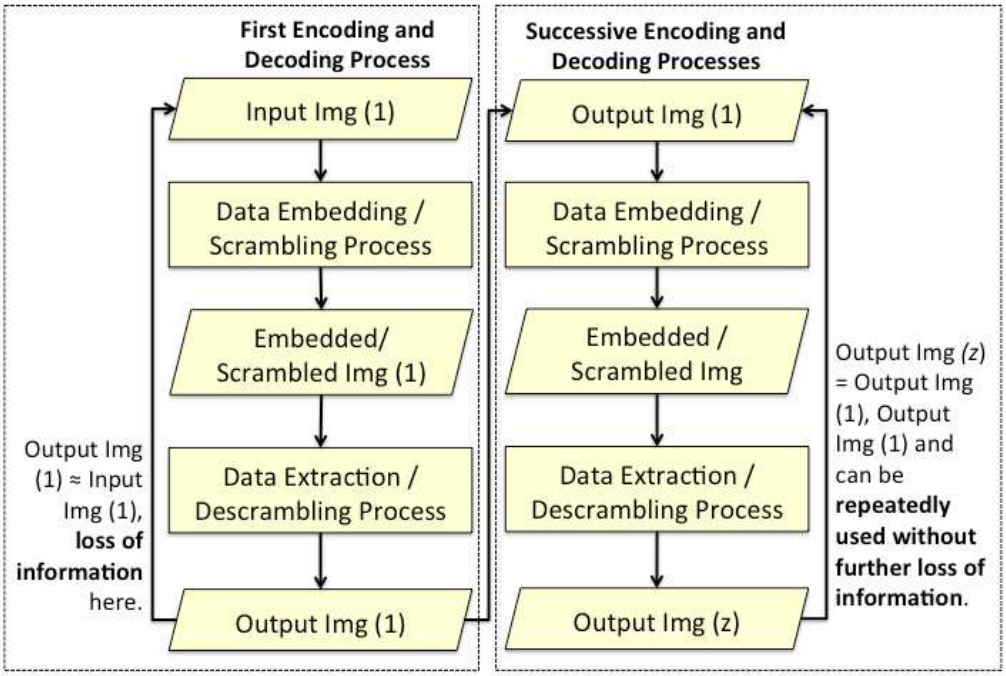


Figure 2.10: Rewritability in Data Insertion

Rewritable is a special case of reversible (as shown in **Figure 2.10**). In rewritable mode, information lost only during the first embedding or scrambling process. The carrier can be reutilized again for successive data insertion or scrambling without causing further loss of information.

2.4.2 Scalability

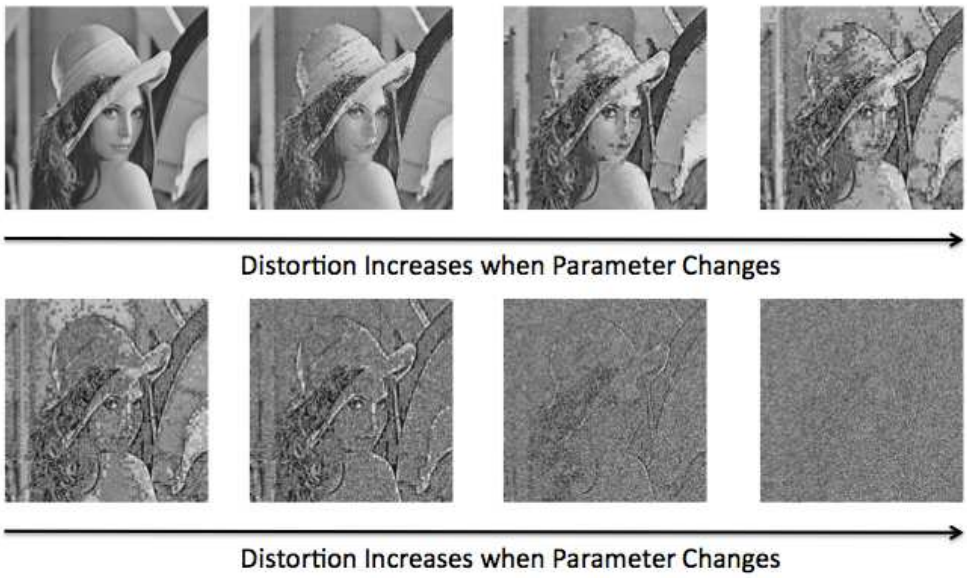


Figure 2.11: Scalability in Image Quality Degradation

From the perspective of data encryption, researchers consider scalable scrambling (also known as partial scrambling, progressive quality degradation or tunable encryption) techniques (Y. Wang, O'Neill, & Kurugollu, 2013; Wong & Tanaka, 2010a; Takayama, Tanaka, Nakajima, & Kouchi, 2008; Massoudi, Lefebvre, De Vleeschouwer, Macq, & Quisquater, 2008; S. Li, Chen, Cheung, Bhargava, & Lo, 2007; C. Wang, Yu, & Zheng, 2003; Dufaux & Ebrahimi, 2008; Pazarci & Dipcin, 2002; Cheng & Li, 2000) to offer the flexibility in controlling the distortion level of a visual content to reveal the desired level of details. **Figure 2.11** shows an example of scalable scrambling, where the distortion changes along with the parameter.

Conventionally, scrambling aims to distort the image while data embedding strives to preserve the image quality. In unified method, the requirement on high output image quality can be relaxed and thus granting some degree of flexibility. This flexibility enables the scalability in image quality degradation using unified method (Wong & Tanaka, 2010a; Takayama et al., 2008; Tian & Wells, 2004). Capitalizing from the relaxation of image quality, scalability in embedding capacity can also be achieved in unified method.

Scalability in image quality degradation is useful for hierarchical access control, multilevel content protection, etc. Different quality degradation is applied depending on the granted access level (Y.-C. Zeng, Huang, & Liao, 2011). For example, in image / video warehouse system, vendor only needs to store a copy of image / video to produce image / video of different quality depending on the status of the viewer (e,g., owner, buyer, registered user, unregistered user). From the viewer perspective, only one download is needed to preview the image / video, and after the user pays for the particular image / video, a high quality version will be presented to the user.

2.4.3 Commutative

Commutative (also referred to as separable) is a useful property in UIH. Scrambled-embedded image can be decoded without concerning about the order (i.e., either de-scrambling first or data extraction first) to reconstruct the original image and extract the embedded information. In some cases, researchers consider the commutative property so that data embedding is independent from data encryption (i.e., they do not interfere one and another) (Fujiyoshi, 2013; Jose & Abraham, 2013; X. Zhang, 2012; Lian, Liu, Ren, & Wang, 2007; Kundur & Karthik, 2004).

2.5 Potential Applications of UIH

Single-purposed application is no longer sufficient to satisfy the needs of multiple protections and functionalities in the real world scenario. For instance, the scrambling feature can be utilized in storage to avoid unauthorized viewing while a fragile watermark can be inserted to verify the integrity of the scrambled content without revealing any part of the original content. Unified method is also useful in digital right management as justified by Kundur et al. (Kundur & Karthik, 2004), where the scrambling process ensures secured transportation and a fingerprint is inserted to trace illegal distributor. For the same purpose, Zhang et al. and Cancellaro et al. proposed commutative data insertion meth-

ods in encrypted image in the spatial (X. Zhang, 2012) and wavelet (Battisti et al., 2009) domains, respectively. Bouslimi et al. proposed a joint watermarking and encryption algorithm using substitution technique and AES (Advanced Encryption Standard) (Bouslimi, Coatrieux, & Roux, 2011) for secure sharing and transmission of medical image. There are other possible applications that can be implemented using the unified information hiding method. Here, without loss of generality, three possible applications are detailed.

First, the application of unified information hiding can be illustrated by a typical example in the office management scenario involving three roles, namely, manager, secretary and irrelevant personnel. The confidential document (e.g., meeting minutes, report, proposal, design draft) is encrypted and stored in the database to avoid unauthorized viewing by irrelevant personnel. However, it is challenging for a secretary to manage and categorize these confidential documents in the database because she does not have access to the plaintext of the encrypted documents. Therefore, through data insertion, metadata can be added to the encrypted confidential documents to allow the secretary to better manage them. Here, two keys are involved, that is, κ_1 and κ_2. The secretary only knows the key κ_1 to retrieve the metadata while the manager has both keys κ_1 and κ_2, where the latter is utilized to decrypt the encrypted confidential documents. Meanwhile, the irrelevant personnel who has neither κ_1 nor κ_2 cannot access to any part of the confidential document. Therefore, unified information hiding method is able to provide controlled access to the users.

The second possible application capitalizes on the scalability property in quality degradation and embedding capacity. Scalability is useful for image warehouse and video-on-demand services. Conventionally, the seller has to store multiple versions of an image / video in the database to serve different users, e.g., low quality for unregistered user, medium quality for registered user and high quality for buyer. User also has

to download multiple versions of the image / video based on their privileges. From the perspective of a seller, scalability is advantageous because a single image / video in the database can be processed to generate different degree of distortion prior to transmission. For the buyer, only one download is needed for both preview and owning, i.e., the user can access to different versions of the image / video based on the provided key.

Another possible application is the use of scrambling as a camouflage to the embedded data (for S2E approach only), i.e., to divert the attention to scrambling. In S2E approach, the generated output can be made to resemble noise. When an attacker encountered an encrypted (i.e., totally distorted) document, the attention will be channeled towards obtaining the plaintext, i.e., the decrypted image. Thus, the attacker will focus on the de-scrambling process of the image and likely to neglect the embedded message, which was achieved by using the scrambling process. In addition, a key can be further utilized to randomly assign the encoded bit sequence for every permutation. Hence, even if the attacker knows that a message is embedded in the scrambled image, the embedded message cannot be extracted directly without the valid key.

2.6 Summary

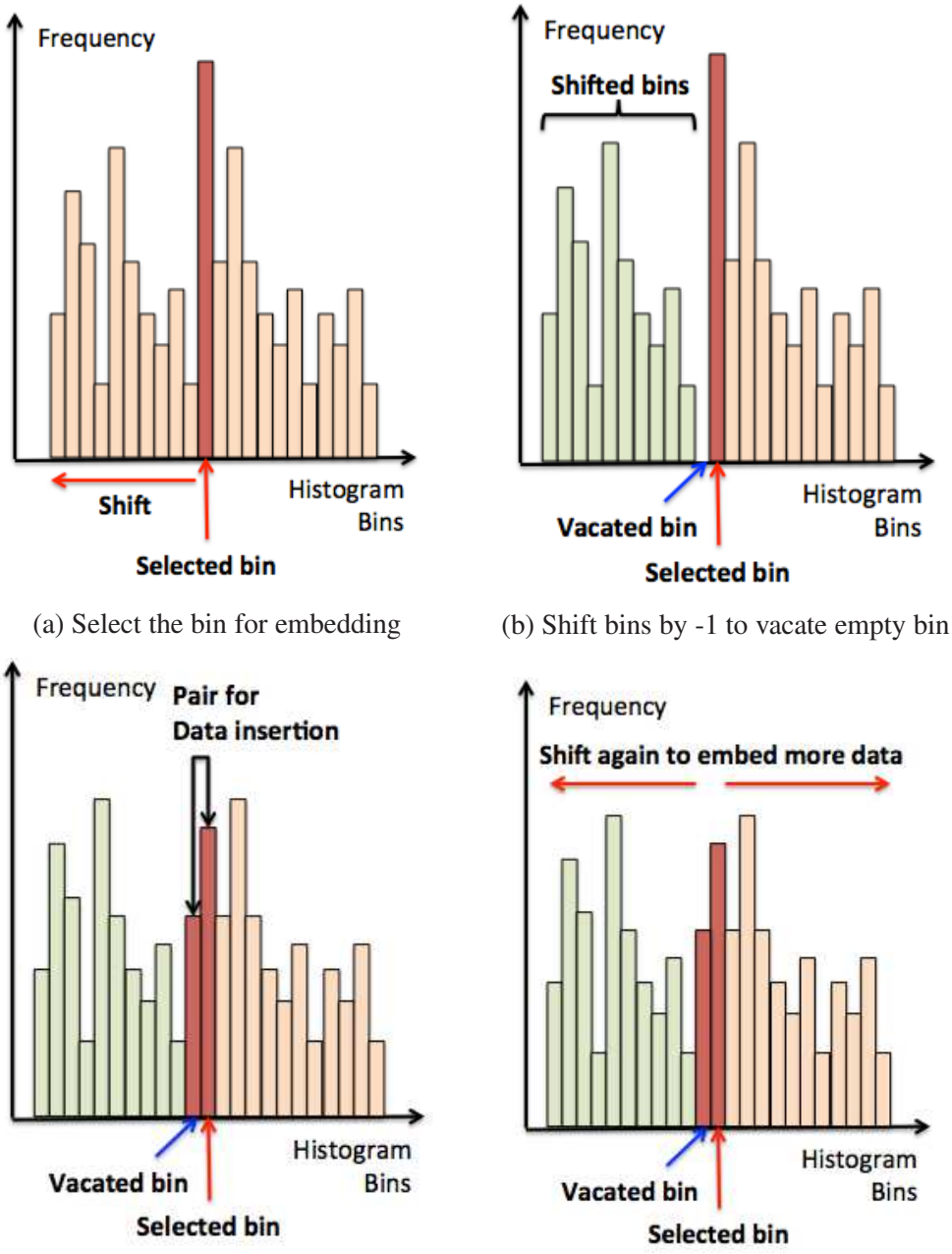
This chapters proposes a general framework to capture difference approaches for achieving UIH. Single-purposed methods (i.e., scrambling methods and data insertion methods) are also reviewed. Related works for each approaches (i.e., ETS, STE, E2S, and S2E) are also discussed and analyzed. Lastly, important properties in unified information hiding and the potential applications are presented.

CHAPTER 3

UIH USING HISTOGRAM ASSOCIATION MAPPING (HAM) IN SPATIAL DOMAIN

This chapter proposes a novel reversible information hiding method aiming to achieve scalable embedding capacity while progressively distorting the image quality following the E2S approach. Unlike the conventional methods, the proposed Histogram Association Mapping (HAM) method purposely degrades the perceptual quality of the input image through data embedding. HAM eliminates the expensive pre-processing step(s) required by the conventional histogram shifting based embedding approach and improves its embedding capacity. In particular, the host image is divided into non-overlapping blocks and each block is classified into two classes. Each class undergoes different HAM process to embed the external data while distorting quality of the image to the desired level. Experiments were conducted to measure the performances of the proposed method by using standard test images and the CalTech 101 dataset (Fei-Fei, Fergus, & Perona, 2007). The proposed method is applicable to content management scenario where an inferior officer (e.g., clerk, nurse) can extract the inserted information from the perceptually degraded (i.e., scrambled) image to administer the file (e.g., copy, archive, move, etc.). On the other hand, the superior officer (e.g., managing director, doctor) can have access to both the external information and the original image.

The survey on conventional histogram method are presented in **Section 3.1** and the specific objectives of this chapter are presented in **Section 3.2**. The fundamental of the proposed method is detailed in **Section 3.3** while **Section 3.4** elaborates the enhancements of the proposed method. Additional properties of the proposed methods are discussed in **Section 3.5**. Experimental results are presented in **Section 3.6** and **Section 3.7** summa-



(a) Select the bin for embedding

(b) Shift bins by -1 to vacate empty bin

(c) Data insertion using selected bin and vacated bin

(d) Shift for more embedding capacity

Figure 3.1: Conventional Histogram Shifting Encoding Process

rizes this chapter.

3.1 Introduction and Related Works

Histogram shifting (HS) is one of the popular data insertion methods, which create empty bins in the image histogram to embed payload (Jung, Ha, & Ko, 2011; Xuan et al., 2009; Ni et al., 2006). In particular, the traditional HS methods are implemented as

shown in **Figure 3.1(a) - (d)**. The process can be illustrated in two stages, namely, (1) select bin for data embedding (**Figure 3.1(a)**) and vacating room for data embedding by shifting histogram bins (**Figure 3.1(b)**), and; (2) mapping values from the selected bins based on the information to be embedded (**Figure 3.1(c)**). If the selected bins are not sufficient to accommodate all the external data, the shifting process will be re-applied (**Figure 3.1(d)**). This class of data embedding methods is reversible and straight forward in implementation. However, the embedding process involves expensive pre-processing (i.e., increasing or decreasing magnitude for pixels of value above or below the selected bin, respectively) and the embedding capacity is still relatively low. Besides, the capacity depends on the distribution of pixel values (i.e., histogram). In the worst case scenario, if all the histogram bins are occupied, no external data can be embedded. In addition, histogram shifting method also requires additional treatment to prevent the underflow and overflow problems.

3.2 Objectives

The main objective of this chapter is to propose a histogram manipulation method (namely, HAM) on image using the E2S approach while progressively distorting its perceptual quality and achieving scalable embedding capacity. Instead of aiming to maintain high image quality, the proposed method purposely distorts the host image to hide its perceptual meaning. In addition, the following objectives pertaining to the problems of conventional histogram shifting method will also be addressed:

1. Eliminate the expensive preprocessing,
2. Mitigate overflow and underflow problems,
3. Achieve higher embedding capacity, and
4. Embed data when all bins in the histogram are occupied.

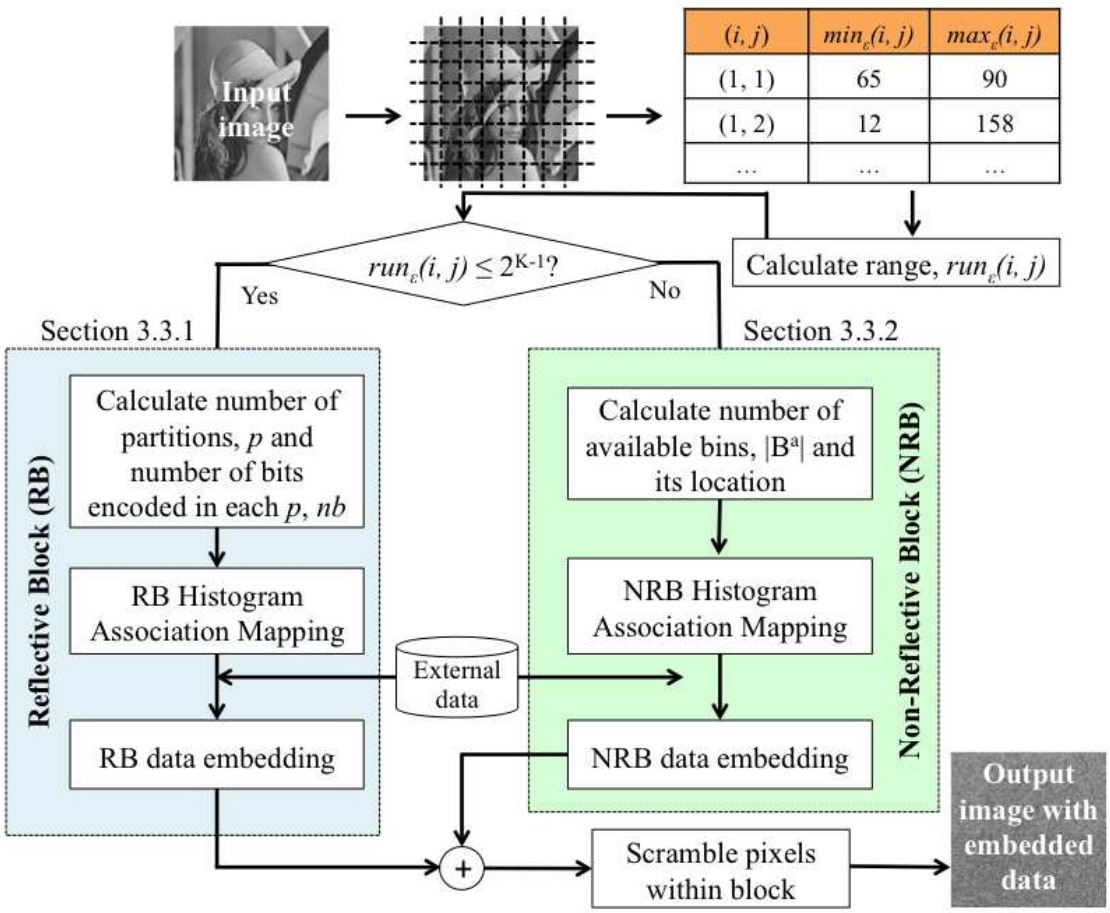


Figure 3.2: Process flow for the proposed method

3.3 Proposed Method

This section details the HAM as a data representation scheme to encode information and its flow is shown in **Figure 3.2**. The input image G with K-bit depth is divided into non-overlapping blocks $B_\varepsilon(i,j)$ each with size of $b_{\varepsilon_1} \times b_{\varepsilon_2}$ pixels, where i and j together denote the location of the block. To ease the discussion, we let $b_{\varepsilon_1} = b_{\varepsilon_2} = b_\varepsilon$. For every block, the maximum pixel value and minimum pixel value are obtained and stored as $\max_\varepsilon(i,j)$ and $\min_\varepsilon(i,j)$, respectively. The range of value $r_\varepsilon(i,j)$ in $B_\varepsilon(i,j)$ is computed as follows:

$$r_\varepsilon(i,j) = \max_\varepsilon(i,j) - \min_\varepsilon(i,j) + 1. \tag{3.1}$$

The blocks are further classified into two categories by using $r_\varepsilon(i,j)$. In particular,

if $r_\varepsilon(i,j) \leq 2^{K-1}$, the block is categorized as RB (Reflective Block). Otherwise, the block is categorized as NRB (Non-Reflective Block). The RB's and NRB's are handled differently to realize data embedding. In specific, the histogram $H_\varepsilon(i,j)$ of each block $B_\varepsilon(i,j)$ is constructed. The bins in the range of $[\min_\varepsilon(i,j), \max_\varepsilon(i,j)]$ in RB and NRB are called *occupied bins* and they are denoted by $B^o_\varepsilon(i,j)$. On the other hand, the bins in the range $[0, \min_\varepsilon(i,j)-1] \cup [\max_\varepsilon(i,j)+1, 2^K - 1]$ are referred to as *available bins* and they are denoted by $B^a_\varepsilon(i,j)$.

3.3.1 Data Embedding in Reflective Blocks (RB)

In this section, Histogram Association Mapping and data embedding processes in RB are detailed. Initially, the number of partitions $p_\varepsilon(i,j)$ which can be formed within $H_\varepsilon(i,j)$ is calculated using **Equation (3.2)**.

$$p_\varepsilon(i,j) = 2^K / \lceil 2^{\log_2 r_\varepsilon(i,j)} \rceil = 2^{K - \lceil \log_2 r_\varepsilon(i,j) \rceil}, \tag{3.2}$$

where $\lceil z \rceil$ rounds z to the nearest integer towards the positive infinity direction. Next, the number of bits that can be represented in each partition $nb_\varepsilon(i,j)$ is calculated using the equation as follows:

$$nb_\varepsilon(i,j) = K - \lceil \log_2 r_\varepsilon(i,j) \rceil. \tag{3.3}$$

We shall drop the ε, i and j notation when there is no confusion. For the ease of implementation, we have forced the partition size to be the value of power of 2 as suggested by **Equation (3.2)**. The original partition refers to the partition that includes the entire range of values in the original histogram. The other partition(s) is (are) called reflective partition(s). Note that it is possible to have more than one reflective partition. Every bin in the original partition is mapped to a bin in at least one reflective partition.

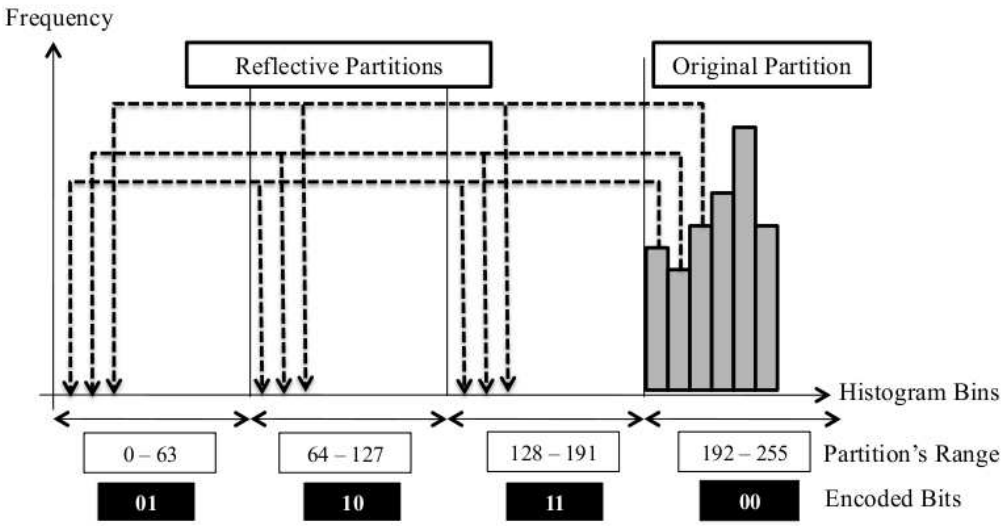


Figure 3.3: An illustration of Histogram Association Mapping in RB when $r = 59$ ($p = 4$ and $nb = 2$)

This mapping process is referred to as HAM and it can be expressed as follows:

$$\text{HAM} : [\min, \max + \delta] \rightarrow [0, \min - 1] \cup [\max + \delta + 1, 2^K - 1], \tag{3.4}$$

where $\delta \geq 0$ so that $\max + \delta - \min = 2^z$ for some z.

With this representation scheme, $| x - x' | \geq r$ holds true for $x \in [\min, \max + \delta]$ and $x' \in [0, \min - 1] \cup [\max + \delta + 1, 2^K - 1]$. In other words, the error is larger than or equal to r when the modification is performed during data embedding.

Figure 3.3 illustrates an example (using an 8-bit image) with min = 192 and max = 250, which results in 3 reflective partitions. Here, the total number of partitions (denoted by p) is 3 + 1 = 4. Hence, '00', '01', '10' and '11' are assigned to each partition as shown. Bin of value 192 in the original partition (hereinafter $b(192)$) is associated with 3 bins in the reflective partitions, i.e., $b(0)$, $b(64)$ and $b(128)$. For data embedding purpose, the pixel value is modified to its corresponding reflective bin value according to the external data to be inserted. For instance, if '10' is to be embedded, the pixel value of 192 is modified to 64.

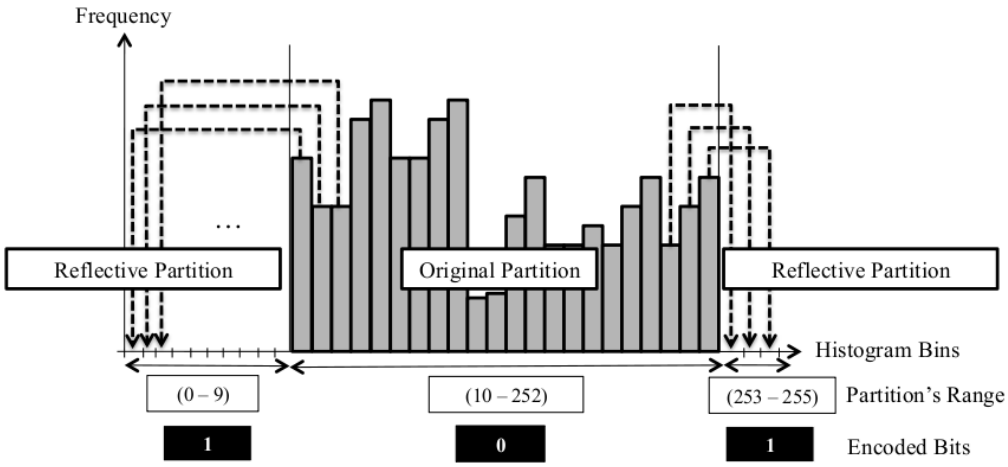


Figure 3.4: An illustration of Histogram Association Mapping in NRB when $r = 243$ ($\mid B^a \mid = 13$)

3.3.2 Data Embedding in Non-Reflective Blocks (NRB)

This section describes HAM in NRB. The number of available bin in NRB can be calculated as:

$$\mid B^a \mid = 2^K - r, \tag{3.5}$$

where $\mid X \mid$ denotes the cardinality of the set X. Here, the histogram H is divided into two partitions only (i.e., one original partition and one reflective partition). Bins in the original partition are associated with bins in the reflective partition B^m. **Figure 3.4** illustrates such an association for the case of max = 252 and min = 10. On the left hand side of the histogram, HAM is captured by the following expression:

$$x' \leftarrow \begin{cases} x & \text{if '0' is to be embedded;} \\ x - \min & \text{otherwise,} \end{cases} \tag{3.6}$$

where x is the original pixel value and x' is the value with embedded data. Similarly, the

association on the right hand side of the histogram is captured by the expression:

$$x' \leftarrow \begin{cases} x & \text{if '0' is to be embedded} \\ x + \max & \text{otherwise} \end{cases} \tag{3.7}$$

For NRB, $| B^a |$ has the same size with $| B^m |$ because every bin in B^a is associated with one bin in B^o, i.e., one-to-one relation. In addition, $| B^m | < | B^o |$ holds true for NRB by definition. On the contrary, $| B^a | \geq | B^o |$ and the mapping from B^o to B^a can be one-to-many, leading to the opportunity to achieve higher embedding capacity.

Using the associated bins of $b(12) \in B^o$ and $b(2) \in B^a$ in **Figure 3.4**, the pixel value remains to be 12 if the external data is '0'. Otherwise, the pixel value is modified to 2 (i.e., its associated bin value in B^a) to embed '1'.

The processed RB's and NRB's (i.e., manipulated blocks output from HAM and embedding processes) are combined to produce an output image G^e with embedded data. Since the distortion caused by data embedding in RB and NRB are different, G^e scrambling (i.e., shuffling the pixels by changing their locations) within non-overlapping blocks of equal size each with $b_{s_1} \times b_{s_2}$ pixels is carried out to regulate the uneven distortion in G^e. To facilitate the discussion, we let $b_{s_1} = b_{s_2} = b_s$. The purpose here is to ensure that the distortion level is almost the same for each block. Finally, the final output, i.e., image embedded with external data and scrambled, is produced. This output is denoted by G'.

Note that each block $B_\varepsilon(i, j)$ of the host image is distorted by a difference (i.e., $| x' - x |$) of at least $r_\varepsilon(i, j)$ and $\min\{\min_\varepsilon(i, j), 2^K - 1 - \max\}$ for the case of RB and NRB, respectively. Notably, most of the external data insertion algorithms aim to maintain the perceptual quality but the proposed method purposely distorts the host image. In particular, the proposed method is able to distort the host image in a progressive manner using different values of b_ε and b_s.

3.3.3 Decoding Process in RB and NRB

Image G' is first divided into $b_s \times b_s$ non-overlapping blocks and descrambled (i.e., recover pixels' original position) to obtain G^e. Next, G^e is divided into $b_\varepsilon \times b_\varepsilon$ non-overlapping blocks. The side information is retrieved and utilized to identify the range and category for every block using **Equation (3.1)** while assuming that $\min_\varepsilon(i,j)$ and $\max_\varepsilon(i,j)$ are known from the side information discussed later.

For RB, $p_\varepsilon(i,j)$ and $nb_\varepsilon(i,j)$ are identified to locate the original and reflective partition(s). With the gathered information, the embedded data can be extracted and the original pixel values can be recovered. For the example shown in **Section 3.3.1**, the recovered side information is min = 192 and max = 250. Then, it is identified that $b(192)$ in the original partition is associated with $b(0), b(64)$ and $b(128)$ in the reflective partitions using **Equation (3.4)**. Hence, if the modified pixel is of value 64, the embedded data is '10' and the pixel value 64 is restored to the original pixel value of 192.

For NRB, $\mid B^a(i,j) \mid$ and the associated bins (left and right associations) are obtained by using **Equation (3.4)**, **(3.6)** and **(3.7)**. For the same example given in **Section 3.3.2** where min = 10 and max = 252, $b(12)$ in the original partition is associated with $b(2)$ in the reflective partition. If the modified pixel is of value 2, the embedded data is '1' and the value 2 is restored to 12 for image recovery purposes.

These processes are repeated for all RB and NRB blocks to extract the embedded data and reconstruct the original image. Since we can deterministically identify the original value for every pixel processed by the proposed HAM, the original image can be perfectly reconstructed. Hence, the proposed method is reversible.

3.4 Enhancements in HAM

In this section, we improve the embedding capacity of the proposed method by modifying the Histogram Association Mapping to exploit all available bins and fully utilize

bins with high frequency of occurrences.

3.4.1 Enhancement in RB

In **Section 3.3**, for each $x \in B^o$, the number of possible values $y \in B^a$ which can be associated with x is restricted to 1, 3, 7, 15, $\cdots$. In other words, the number of partitions is restricted to power of 2. Hence, wastage of embedding venues is inevitable. For example, suppose that r = 65 and by definition it is an RB. The original partition will be of size 128. The number of unutilized bins $\mid B^u \mid$ can be calculated as:

$$\mid B^u \mid = (2^w - r) \times p, \tag{3.8}$$

where $w = \lceil \log_2(r) \rceil$, which is the size of the original partition with wastage. Using **Equation (3.8)**, $\mid B^u \mid$ is (128 - 65) $\times$ 2 = 126 bins when r = 65, which translates to $\sim$49% of unutilized bins in the histogram.

Equation (3.3) determines the number of bits that can be held by each pixel (i.e., embedding capacity) in the block under consideration, which is the cause of wastage problem. Instead of using **Equation (3.3)** that assigns uniform embedding capacity to each bin in B^o, a non-uniform approach is proposed. In particular, each bin in B^o will be assigned to host different amount of information. To ease the discussion, the following sets are defined:

$$\Gamma(k) = \{(l, \alpha_l, \beta_l) : 0 \leq l \leq k\}, \tag{3.9}$$

for $k = K, K-1, \cdots, 1$. Here, each triplet in a specific $\Gamma(k)$ implies that the first α_l bins in B^o will be utilized to each encode l bits. In other words, each of these α_l bins will be associated with $2^l - 1$ bins in B^a. In the similar manner, the next α_{l-1} bins in B^o will be utilized to encode $l-1$ bits, and so forth. Here, $\beta_l = \alpha_l \times (2^l - 1)$ and the following holds

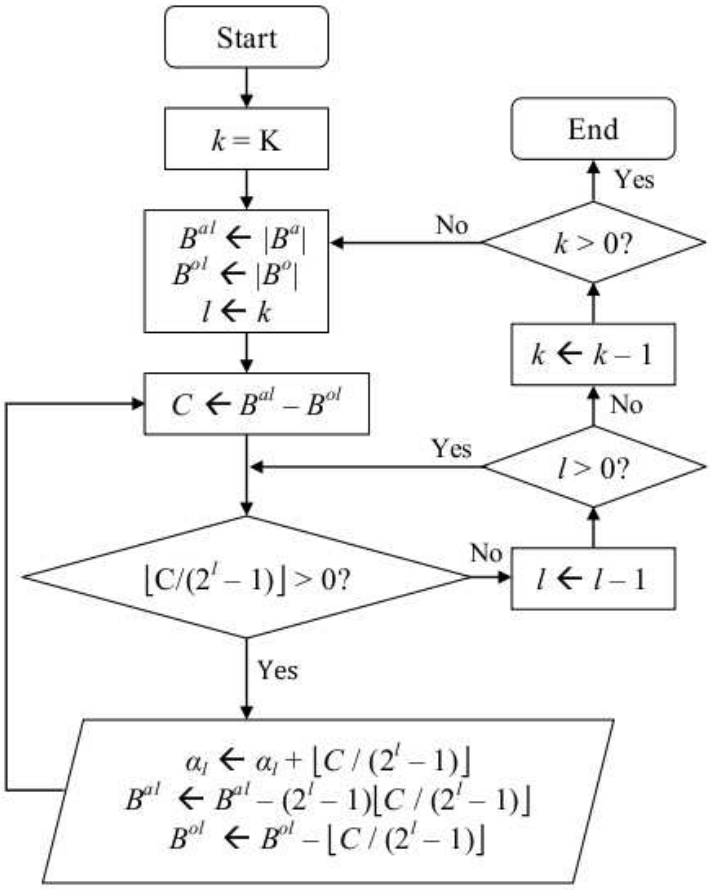


Figure 3.5: Algorithm to search for (l, α_l, β_l) for various k in the proposed enhancement for RB

true for all k:

$$r = \sum_{l=1}^{k} \alpha_l. \tag{3.10}$$

Here, we ensure that:

$$\sum_{l=1}^{k} 2^l \times \alpha_l \leq \sum_{l=1}^{k} \alpha_l + \beta_l \leq 2^K, \tag{3.11}$$

in which for any given k, we want to maximize the value of α_l.

Figure 3.5 shows the process flow of the proposed algorithm in finding the set of triplets $\Gamma(k)$ for various k that offers the highest embedding capacity. While doing so, it is ensured that each bin in B^o is modified (i.e., associated with a value outside of the range $[\min, \max]$) during the data embedding process so that distortion is achieved.

Table 3.1: Snapshots of the input and output for the RB enhancement method with $r = 65$ and $k = 5$

Iteration	Input	Output	(l, α_l, β_l)
1	$B^{ol} = 65$ $B^{al} = 191$ $\underline{l = 5}$	B^{ol} =61 B^{al} =67 $l = 5$ $\alpha_5 = 4$ $\beta_5 = 124$	(5, 4, 124)
2	$B^{ol} = 61$ $B^{al} = 67$ $l = 5$ $\alpha_5 = 4$ $\beta_5 = 124$	$B^{ol} = 61$ $B^{al} = 67$ $\underline{l = 4}$	-
3	$B^{ol} = 61$ $B^{al} = 67$ $l = 4$	$B^{ol} = 61$ $B^{al} = 67$ $\underline{l = 3}$	-
4	$B^{ol} = 61$ $B^{al} = 67$ $l = 3$	$B^{ol} = 61$ $B^{al} = 67$ $\underline{l = 2}$	-
5	$B^{ol} = 61$ $B^{al} = 67$ $l = 2$	$B^{ol} = 59$ $B^{al} = 61$ $l = 2$ $\alpha_2 = 2$ $\beta_2 = 6$	(2, 2, 6)
6	$B^{ol} = 59$ $B^{al} = 61$ $l = 2$ $\alpha_2 = 2$ $\beta_2 = 6$	$B^{ol} = 59$ $B^{al} = 61$ $\underline{l = 1}$	-
7	$B^{ol} = 59$ $B^{al} = 61$ $l = 1$	$B^{ol} = 0$ $B^{al} = 2$ $l = 1$ $\alpha_1 = 59$ $\beta_1 = 59$	(1, 59, 59)
8	$B^{ol} = 0$ $B^{al} = 2$ $l = 1$	$B^{ol} = 0$ $B^{al} = 2$ $\underline{l = 0}$	(0, 0, 2)

Table 3.1 records the snapshots of each iteration of the proposed algorithm for $r = 65$ and $k = 5$. Note that the value of l decreases from $k = 5$ to 0 and it could remain unchanged for more than one iteration. Again, we emphasize that the purpose here is to search for the largest α_l while ensuring that each pixel value in B^o is associated with at least one value in B^a. The list of triplets $\{(l, \alpha_l, \beta_l) \mid l = k, k-1, \cdots, 0\}$ (recorded in the right most column) will be utilized to guide the embedding process.

Here, we step through the proposed algorithm in **Figure 3.5** using the example shown in **Table 3.1**. Initially, l is set to 5 since $k = 5$. The leftover occupied bins B^{ol} is 65 and

Table 3.2: Output from searching process for various k by using **Figure 3.6** when r = 65

k	$\Gamma(k)$
8	(6, 2, 126), (1, 63, 63), (0, 0, 2)
7	(6, 2, 126), (1, 63, 63), (0, 0, 2)
6	(6, 2, 126), (1, 63, 63), (0, 0, 2)
5	(5, 4, 124), (2, 2, 6), (1, 59, 59), (0, 0, 2)
4	(4, 8, 120), (3, 2, 14), (1, 55, 55), (0, 0, 2)
3	(3, 20, 140), (2, 2, 6), (1, 43, 43), (0, 0, 2)
2	(2, 62, 186), (1, 3, 3), (0, 0, 2)
1	(1, 65, 65), (0, 0, 126)

leftover available bins B^{al} is 191. The usable available bins (denoted by C) is calculated as $C = B^{al} - B^{ol}$ to ensure that all the B^{o} are considered and in this case, C is 126. For each l, $\alpha_l = \lfloor C/(2^l - 1) \rfloor$ and $\beta_l = \alpha_l \times (2^l - 1)$ are computed. In the first iteration (i.e., $l = 5$), $\alpha_5 = 4$ and $\beta_5 = 124$. For the second iteration, the same l (i.e., 5) is utilized but there are not enough available bins that can be utilized to associate with the occupied bins. Therefore, l is decreased by one and the process continues until all the occupied bins are considered (i.e., $l = 0$).

Table 3.2 shows the outputs of the searching process for various k when r = 65. The number of unutilized bins (i.e., not associated to any value $x \in [\min, \max]$) is indicated by the last triplet (i.e., β_0 when l = 0) for the output of each k. **Table 3.2** indicates that the number of unutilized bins is 2 for $k > 1$. When k = 1, the triplet (1, 65, 65) implies that there are 65 bins in B^{o} to encode 1 bit and the triplet (0, 0, 126) implies that there are 126 unutilized bins, which is the same as in the original proposed HAM detailed in **Section 3.3**. Therefore, it is obvious that the wastage is reduced in this proposed enhancement and the embedding capacity is hence increased.

Since the frequency of each value in the image varies, $\Gamma(k)$ which produces the highest embedding capacity is utilized for data embedding purposes. In particular, the

Table 3.3: Embedding capacity for various k by using **Table 3.2**

k	1	2	3	4	5	6	7	8
Block Capacity (Bits)	64	121	132	131	**133**	117	117	117
Block Capacity (Bpp)	1.00	1.89	2.06	2.05	**2.08**	1.83	1.83	1.83
Unutilized Bins	126	2	2	2	2	2	2	2

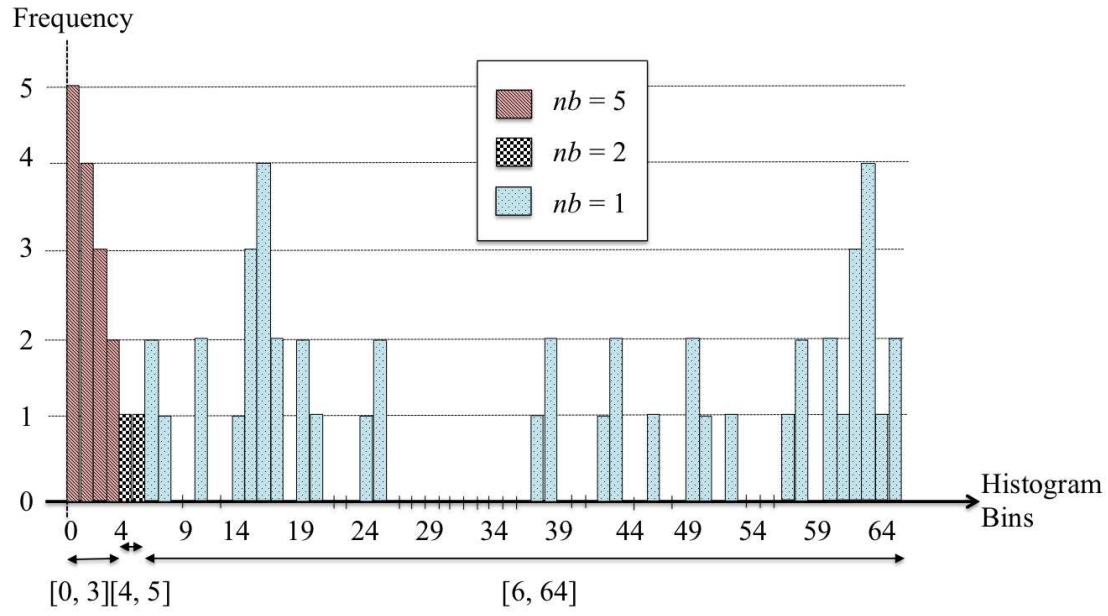
embedding capacity for RB by using various k is calculated as:

$$\sum_{l=1}^{k} l \times \text{Freq}(b(\alpha_l)), \tag{3.12}$$

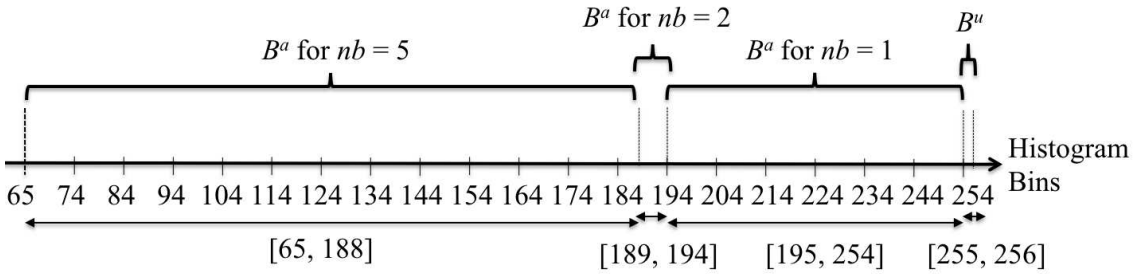
where $\text{Freq}(b(\alpha_l))$ is the total number of pixels in the α_l consecutive bins selected to encode l bits. **Table 3.3** shows the embedding capacity achieved by the block shown in **Figure 3.6** by using **Equation (3.12)** for various starting k (i.e., the starting value). It is observed that larger k does not always offer higher embedding capacity. Nevertheless, by integrating the proposed RB enhancement, embedding capacity can be increased from 1.00 bpp to 2.08 bpp by setting $k = 5$. Therefore, $k_\varepsilon(i, j) = 5$ for this block and it will be output as side information to enable correct message extraction and reversibility purposes.

Figure 3.6 shows the distribution of embedding capacity for $r = 65$ and $k_\varepsilon(i, j) = 5$. Here, the example of $\Gamma(5) = \{(5, 4, 124), (2, 2, 6), (1, 59, 59), (0, 0, 2)\}$, which is selected by the proposed enhancement algorithm, is shown. In particular, pixels in the first 4 bins in B^o will each be utilized to hold 5 bits of data where each of these 4 bins in B^o will be associated with 31 bins in B^a. Pixels belonging to the next 2 bins (i.e., 5th and 6th) are utilized to encode 2 bits each, where $2 \times 3 = 6$ bins in B^a are needed. After that, the rest of the bins in B^o (i.e., 59 bins in total) and 59 bins in B^a are associated to encode 1 bit. Finally, 2 bins are left unutilized.

The data embedding process in the proposed enhancement is similar to that of the original method discussed in **Section 3.3**. For example, $b(4)$ is utilized to encode '00', and it is associated with 3 available bins, namely $b(189)$, $b(191)$ and $b(193)$ to represent



(a) Association in B^o



(b) Association in B^a

Figure 3.6: An illustration of Histogram Association Mapping using enhanced RB method

'01', '10', '11', respectively. The pixel value is modified to its corresponding reflective bin value according to the input. For instance, 4 is modified to 191 if the external data of '10' is to be embedded, and so forth.

Nevertheless, $\max_\varepsilon(i,j)$, $\min_\varepsilon(i,j)$ and $k_\varepsilon(i,j)$ for every block $B_\varepsilon(i,j)$ are stored as side information (detailed in **Section 3.5.1**) for recovery purposes. During decoding, the same table considered in the encoding process (e.g., **Table 3.2**) can be generated after r is calculated using max and min. The value k is utilized to inform the decoder which set of $\Gamma(k)$ was selected during the encoding process. After that, bins in the original partition are associated with the corresponding bins in the reflective partition(s) based on $\Gamma(k)$. Finally, the embedded data is extracted and the modified pixels are restored to their

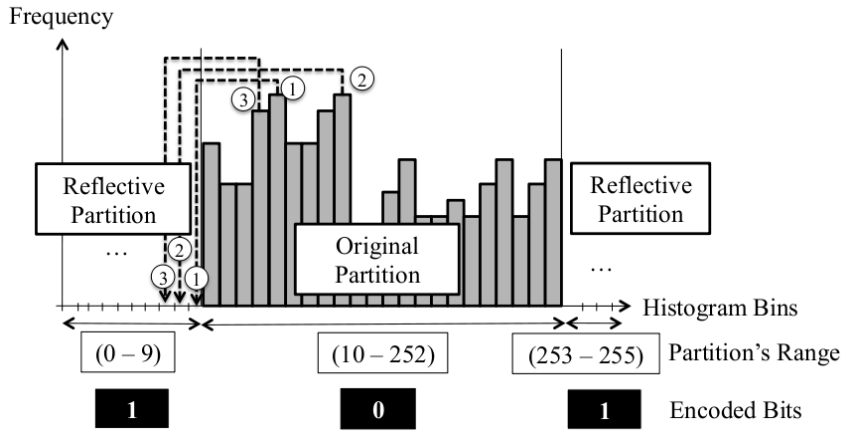


Figure 3.7: Histogram Association Mapping using enhanced NRB method T1

original values in the original partition.

3.4.2 Enhancement in NRB

In the original method, the embedding capacity in a NRB depends on the frequency of occurrences of the selected bins. However, the bins of high frequencies are not always selected for data embedding purposes. Therefore, this section proposes two enhancements in NRB to exploit the frequency feature in the histogram for improving embedding capacity.

Enhancement Technique I (T1) for NRB

The aim here is to choose the values $x \in B^o$ with high frequency of occurrences to associate with $y \in B^a$ for increasing embedding capacity. By definition $B^a = [0, \min -1] \cup [\max +1, 2^K - 1] = \{0, 1, 2, \cdots, \min -1, \max +1, \max +2, \cdots, 2^K - 1\}$ when forcing $\delta = 0$. Let $B^a(z)$ be the z-th available bins in the histogram of the NRB under consideration where $z \in \{1, 2, \cdots, T\}$ and $T = \mid B^a \mid = \min -1 + 2^K - 1 - \max$. $B^a(1)$ is associated with the value in B^o with the highest frequency, $B^a(2)$ is assigned to the value in B^o with the second highest frequency, and so forth. In other words, the T number of the most frequently occuring (bin) values in B^o are associated with the elements in B^a to increase

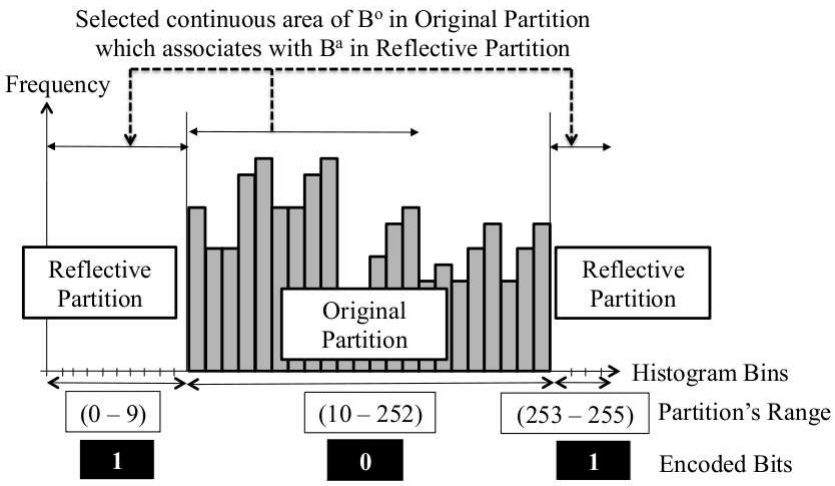


Figure 3.8: Histogram Association Mapping using enhanced NRB method T2

embedding capacity.

Association using T1 is illustrated in **Figure 3.7**. Here, $b(9)$ is associated with $b(14)$. Again, a pixel of value 14 assumes the value 14 (i.e., no change) if '0' is to be embedded, or it is modified to 9 otherwise. Note that side information is needed in T1 for data extraction and reversibility purposes. The side information includes min (8 bits), max (8 bits) and the bin values in B^o selected to embed data (T $\times$ 8 bits).

Enhancement Technique II (T2) for NRB

T2 proposes to search for a continuous interval $[SV, SV+T-1]$ so that it produces the highest cumulative frequency. In other words, it searches for a starting bin value SV as follows:

$$\arg\max_{SV} \left\{ \sum_{z=SV}^{SV+T-1} \text{Freq(t)} \mid \min \leq SV \leq \max -T+1 \right\}. \tag{3.13}$$

Once SV is found, $B^a(z)$ is associated with $b(SV+z)$ for $z \in \{1,2,\cdots,T-1\}$. An example is shown in **Figure 3.8**. Similar to the processes in T1, $b(10)$ is associated with $b(9)$. A pixel of value 10 remains unchanged if '0' is to be embedded or it is modified to 9 otherwise. T2 also requires the storage of side information. In particular, $\min_\varepsilon(i,j)$ (8

bits), $\max_{\varepsilon}(i,j)$ (8 bits) and $SV_{\varepsilon}(i,j)$ (8 bits) are output as side information, which sums up to 24 bits per NRB.

Decoding Process for T1 and T2

For the decoding process in T1 and T2, T is first calculated using $\min_{\varepsilon}(i,j)$ and $\max_{\varepsilon}(i,j)$ retrieved from the side information. Similar to the encoding process, **Equation (3.6)** and **(3.7)** are not needed in T1 because the actual bin values in B^o (i.e., T of them in total) were stored as side information. Thus, the association in T1 is re-established using the side information directly. For T2, $SV_{\varepsilon}(i,j)$ is retrieved from the side information to re-establish the association with the bins in B^a. The rest of the decoding process for NRB follows that of **Section 3.3.3** and we omit the presentation here.

3.5 Discussion

This section discusses the side information needed in the decoding process and robustness of HAM against brute force attack.

3.5.1 Side Information

min	***max***	***k*** (RB) | ***selected bins*** (NRB T1) | ***SV*** (NRB T2)

Figure 3.9: Layout of side information

HAM and its enhancements require side information for correct data extraction and reversibility purposes. The side information required includes max and min for each block and k for the case of RB. In case of NRB, all selected bins for T1 or the values of SV for T2, are stored. To reduce size of the side information, the min and max values are predicted using $\min_{\varepsilon}(i,j) = \min_{\varepsilon}(i,j-1)$ and the errors are entropy coded using GRC (Golomb-Rice Coding) (Golomb, 1966). The layout of the side information is shown in

Figure 3.9. The number of bits required to encode k is $\lceil \log_2 K \rceil$, where K is the bit depth of the image.

3.5.2 Robustness against Brute Force Attack

Here, we discuss the total number of possible combinations (i.e., trials) that needs to be considered in the brute force attack to extract the embedded data and reconstruct the original image perfectly.

For the basic HAM method proposed in **Section 3.3**, the number of possible combination is

$$\prod_{q=1}^{4} F(C_q), \tag{3.14}$$

where C_q are defined as follows:

$$\begin{aligned} 8M \times 8N \text{ combinations for } b_{s_1} \text{ and } b_{s_2} &: F(C_1) \\ 8M \times 8N \text{ combinations for } b_{\varepsilon_1} \text{ and } b_{\varepsilon_2} &: F(C_2) \\ (2^K)^{\frac{8M \times 8N}{b\varepsilon_1 \times b\varepsilon_2}} \text{ combinations for } \min(i,j) \text{ in all blocks} &: F(C_3) \\ (2^K)^{\frac{8M \times 8N}{b\varepsilon_1 \times b\varepsilon_2}} \text{ combinations for } \max(i,j) \text{ in all blocks} &: F(C_4), \end{aligned} \tag{3.15}$$

and $8M \times 8N$ denotes the dimension of the image. In addition, simple permutations can be added to further complicate the brute force attack. In particular, prior to the encoding process, the pixels in each $b_{\varepsilon_1} \times b_{\varepsilon_2}$ block can be randomly permuted using a key. The position of the pixels in each $b_{s_1} \times b_{s_2}$ block can also be permuted using another key as described in **Section 3.3** to even out the distortion after the encoding process. The number of possible combinations for these scrambling techniques are captured by $F(C_5)$:

$$F(C_5) = \prod_{i=1}^{8M/b_{\varepsilon_1}} \prod_{j=1}^{8N/b_{\varepsilon_2}} c(i,j),$$

where

$$c(i,j) = \begin{cases} (b_{s_1}(i,j) \times b_{s_2}(i,j))! & \text{if } b_{s_1}(i,j) \times b_{s_2}(i,j) > b_{\varepsilon_1}(i,j) \times b_{\varepsilon_2}(i,j) \\ (b_{\varepsilon_1}(i,j) \times b_{\varepsilon_2}(i,j))! & \text{otherwise.} \end{cases} \tag{3.16}$$

Furthermore, $F(C_6)$ indicates the number of possible combinations to randomly associate the available bins to the occupied bins instead of using **Equation 3.4**, **3.6** and **3.7**. The representation of the encoded bits can also be permuted to complicate the extraction process and the number of possible combinations is expressed in $F(C_7)$:

$$\begin{aligned} &| B^a_\varepsilon(i,j) |! \text{ combinations for associating bins } :F(C_6) \\ &p_\varepsilon(i,j)! \text{ combinations for encoded bits } :F(C_7). \end{aligned} \tag{3.17}$$

Since $\prod_{q=1}^{7} F(C_q)$ appears to be a large number, we conclude that the proposed method is robust against brute force attack.

The enhancements for HAM detailed in **Section 3.4** further improves the robustness of the proposed method. Based on the algorithm depicted in **Figure 3.5**, elements (i.e., triplets) of the set $\Gamma(k)$ is updated according to $k_\varepsilon(i,j)$ and the statistic of the block under consideration. Thus, $k_\varepsilon(i,j)$ of $\Gamma(k)$ can also be exploited to enhance the robustness of the proposed method. In particular, instead of searching for $k_\varepsilon(i,j)$ that offers the highest capacity, $k_\varepsilon(i,j)$ can be chosen based on the sequence generated by a pseudo-random number generator seeded by a key. The number of possible combinations for $k_\varepsilon(i,j)$ in RB is captured by $F(C_8)$:

$$K \text{ combinations for } k_\varepsilon(i,j) :F(C_8). \tag{3.18}$$

For the enhancement T1, $F(C_9)$ shows the number of possible combinations to locate all selected bins in $B^o_\varepsilon(i,j)$ (i.e., same size as $\mid B^a_\varepsilon(i,j) \mid$):

$$\mid B^a_\varepsilon(i,j) \mid ! \text{ combinations for selected bins in NRB using T1 : } F(C_9). \tag{3.19}$$

On the other hand, $F(C_{10})$ is the number of possible trials for the selected $SV_\varepsilon(i,j)$ in T2:

$$\mid B^o_\varepsilon(i,j) \mid - \mid T_\varepsilon(i,j) \mid \text{ combinations for } SV_\varepsilon(i,j) \text{ in NRB using T2 : } F(C_{10}). \tag{3.20}$$

Note that $F(C_9)$ and $F(C_{10})$ cannot be utilized together. Hence, the unauthorized party has to consider $\prod_{q=1}^{9} F(C_q)$ number of trials (if T1 is considered) or $F(C_{10}) \times \prod_{q=1}^{8} F(C_q)$ number of trials (if T2 is considered) to correctly extract the embedded data and reconstruct the original image. That is, it requires a significant number of trials to perform the brute force attack on the output image generated by the proposed method. Therefore, we conclude that the proposed method is robust against brute force attack.

3.6 Experimental Results

The proposed method is implemented in Matlab using Mac OS version X, 2GHz Intel Core i7 processor, and 4GB 1333 MHz DDR3 memory, to verify its effectiveness in achieving two objectives, namely scalability in embedding capacity and progressive quality degradation. It is verified that the embedded data can be perfectly extracted and the original image can be losslessly reconstructed. Experiments are carried out by using the CalTech 101 dataset (all 8-bit images) (Fei-Fei et al., 2007) which has 3999×2799 and 132×150 pixels as the largest and smallest image dimensions, respectively. Meanwhile, six standard test images (i.e., Airplane, Baboon, Boat, Lake, Lenna and Peppers) (*The USC-SIPI Image Database [online]*, n.d.) are considered for comparison purposes with the existing methods.

Table 3.4: Average gross embedding capacity [bits per pixel] of the proposed method and its enhancements using the Caltech 101 dataset

b_ε (%)	Original		$\oplus$1				$\oplus$2			
	RB	NRB	RB	NRB	Improvement		RB	NRB	Improvement	
1	4.74		5.01		0.27		5.00		0.26	
	4.85	0.47	5.12	0.94	0.27	0.47	5.12	0.57	0.27	0.10
2	3.43		3.74		0.31		3.70		0.27	
	3.83	0.59	4.12	0.95	0.29	0.36	4.12	0.68	0.29	0.09
3	2.92		3.26		0.34		3.22		0.29	
	3.56	0.51	3.87	0.90	0.31	0.40	3.87	0.68	0.31	0.18
4	2.59		2.94		0.35		2.89		0.30	
	3.42	0.45	3.74	0.85	0.32	0.40	3.74	0.67	0.32	0.22
5	2.31		2.66		0.36		2.48		0.17	
	3.30	0.41	3.63	0.80	0.33	0.39	3.63	0.66	0.33	0.25
6	2.11		2.47		0.36		2.43		0.31	
	3.24	0.38	3.57	0.76	0.33	0.39	3.57	0.64	0.33	0.27
7	1.92		2.29		0.36		2.24		0.32	
	3.16	0.35	3.49	0.73	0.33	0.38	3.49	0.63	0.33	0.28
8	1.80		2.15		0.36		2.11		0.31	
	3.16	0.32	3.48	0.71	0.32	0.38	3.48	0.61	0.32	0.29
9	1.58		1.94		0.36		1.85		0.27	
	2.91	0.32	3.24	0.70	0.33	0.38	3.24	0.61	0.33	0.29
10	1.52		1.88		0.36		1.84		0.32	
	3.03	0.29	3.36	0.66	0.33	0.38	3.36	0.59	0.33	0.30
20	0.69		1.05		0.36		1.00		0.31	
	2.60	0.17	2.97	0.54	0.37	0.37	2.97	0.48	0.37	0.30
30	0.52		0.85		0.33		0.80		0.28	
	2.55	0.12	2.83	0.45	0.28	0.34	2.83	0.39	0.28	0.27
40	0.27		0.58		0.31		0.52		0.25	
	1.38	0.09	1.57	0.39	0.18	0.31	1.57	0.33	0.18	0.25
50	0.06		0.33		0.26		0.25		0.19	
	1.86	0.04	1.99	0.28	0.13	0.24	1.99	0.23	0.13	0.19
60	0.06		0.30		0.24		0.25		0.19	
	0.11	0.04	0.12	0.28	0.01	0.24	0.12	0.23	0.01	0.19
70	0.09		0.32		0.23		0.27		0.18	
	0.33	0.04	0.38	0.28	0.05	0.23	0.38	0.22	0.05	0.18
80	0.12		0.34		0.22		0.29		0.17	
	1.00	0.04	1.13	0.26	0.13	0.22	1.13	0.20	0.13	0.17
90	0.20		0.38		0.18		0.34		0.14	
	2.38	0.03	2.57	0.21	0.19	0.17	2.57	0.16	0.19	0.13
100	0.01		0.14		0.13		0.11		0.10	
	0.00	0.01	0.00	0.14	0.00	0.13	0.00	0.11	0.00	0.10

3.6.1 Carrier Capacity

Table 3.4 records the average gross embedding capacity (in the unit of bits per pixel) for the original proposed method, enhanced RB method and two enhanced NRB (referred to as T1 and T2, respectively) methods for various b_ε using the Caltech 101 dataset. For

Table 3.5: Average distribution of RB and NRB [in percentage] in Caltech 101 dataset

$b_\varepsilon(\%)$	RB (%)	NRB (%)
1	97.5	2.5
2	88.2	11.8
3	80.1	19.9
4	72.7	27.3
5	66.7	33.3
6	61.8	38.2
7	57.8	42.2
8	53.4	46.6
9	52.1	47.9
10	47.4	52.6
20	30.5	69.5
30	22.8	77.2
40	11.3	88.7
50	19.2	80.8
60	1.5	98.5
70	4.2	95.8
80	11.0	89.0
90	26.2	73.8
100	0.1	99.9

completion of discussion, the contribution by RB and NRB are explicitly recorded, along with the improvement achieved in the proposed enhancements. Here, b_ε varies from 1% to 100% of the total number of pixels in each test image. Combination of enhanced RB and enhanced NRB T1 is named $\oplus 1$ while combination of enhanced RB and enhanced NRB T2 is referred to as $\oplus 2$. In general, results show that the gross embedding capacity increases when b_ε decreases. This is because when b_ε decreases, the range $r_\varepsilon(i,j)$ of $B_\varepsilon(i,j)$ also decreases and hence more venues (i.e., more RB) can be utilized for data embedding purposes. Thus, it verifies that the proposed method is able to provide scalable functionality in embedding capacity by varying the block size through parameter b_ε. It is observed that the proposed enhancements (i.e., $\oplus 1$ and $\oplus 2$) outperform the original method in terms of gross embedding capacity and this verifies the effectiveness of the enhanced methods in utilizing the available bins and considering bins of higher frequency of occurrences regardless of the value of b_ε. Here, $\oplus 1$ generally has the highest gross embedding capacity among these three methods.

Table 3.5 records the average distribution of RB and NRB in the Caltech 101 dataset

for various block sizes b_ε. In general, results show a trend where the percentage of RB decreases (while the percentage of NRB increases) when b_ε increases, and vice versa. However, there are some irregular patterns when $50\% \leq b_\varepsilon \leq 90\%$. The fluctuation is due to the image division algorithm considered for handling the Caltech 101 dataset. In particular, images in this dataset are of various sizes and thus the concept of percentage (i.e., from 1% to 100%) is deployed to define and vary the block size for data embedding purposes. The utilization of Caltech 101 dataset also demonstrate the flexibility of the proposed method to operate in images with various sizes. For example, an image with 123×120 pixels is divided into 4 blocks if $b_\varepsilon = 90\%$. These four blocks are of sizes 110×108, 110×12, 13×108, and 13×12 pixels. The irregularity in block size causes the changes in categorization process (i.e., type RB or NRB), and eventually leads to the irregular patterns in **Table 3.5**. On the other hand, when an image can be evenly divided (e.g., 512×512 pixel image divided evenly into four 256×256 pixel blocks), the percentage of RB is decreasing monotonically (while the percentage of NRB is increasing monotonically) as b_ε increases, and vice versa. Nevertheless, it is observed that when b_ε is small, the embedding capacity is contributed mainly by RB. On the other hand, when $b_\varepsilon > 30\%$, more than 70% of the embedding capacity is contributed mainly by NRB.

On the other hand, **Table 3.6** records the effective embedding capacity (i.e., gross embedding capacity minus side information) for the proposed method and its enhancements. To ease the comparison process, the improvement achieved by $\oplus 1$ and $\oplus 2$ over the original proposed method (in terms of percentage) are also recorded. When b_ε is small (i.e., 1%), the original method yields the best result because the capacity in $\oplus 1$ and $\oplus 2$ are mostly consumed by the side information. Results indicate that $\oplus 2$ achieves the highest effective embedding capacity among the proposed approaches for $2\% \leq b_\varepsilon \leq 20\%$ after accommodating all the required side information. $\oplus 1$ needs more space for side information storage because every bin (value) in B^o selected for data embedding has to be

Table 3.6: Effective embedding capacity [bits per pixel] and image quality (SSIM and PSNR [dB]) for the proposed method and its enhancements with respect to the Caltech 101 dataset

b_ε (%)	Effective Carrier Capacity					SSIM	PSNR [dB]
	Original	$\oplus 1$	Improvement [%]	$\oplus 2$	Improvement [%]		
1	**1.53**	0.83	-45.52	0.99	-34.85	0.0850	28.0438
2	2.78	2.09	-24.82	**2.88**	3.54	0.1235	28.6276
3	2.63	1.63	-37.96	**2.84**	7.97	0.1428	29.0242
4	2.29	1.27	-44.63	**2.55**	11.05	0.1577	29.3902
5	2.20	1.43	-35.15	**2.48**	12.63	0.1713	29.7153
6	2.04	1.50	-26.46	**2.33**	14.19	0.1839	30.0034
7	1.87	1.51	-19.47	**2.17**	15.95	0.1969	30.2897
8	1.75	1.53	-12.96	**2.05**	17.09	0.2099	30.5597
9	1.54	1.40	-9.10	**1.85**	19.88	0.2239	30.8046
10	1.49	1.46	-2.33	**1.80**	20.68	0.2395	31.0783
20	0.68	0.93	37.79	**0.99**	45.41	0.3809	33.5798
30	0.52	**0.79**	53.37	0.79	53.30	0.3930	33.5130
40	0.27	**0.55**	103.28	0.52	92.19	0.4700	34.8954
50	0.06	**0.29**	367.93	0.25	297.31	0.6091	38.1303
60	0.06	**0.29**	377.85	0.25	303.40	0.6122	36.3589
70	0.09	**0.31**	262.59	0.27	212.60	0.6191	36.7397
80	0.12	**0.33**	175.73	0.29	143.45	0.6414	37.3505
90	0.20	**0.36**	83.52	0.33	68.48	0.6908	38.4184
100	0.01	**0.14**	1208.87	0.11	871.97	0.7866	36.2568

recorded for restoration purposes as discussed in **Section 3.5.1**. Therefore, it is suggested that $\oplus 2$ should be utilized to achieve higher embedding capacity when $2\% \leq b_\varepsilon \leq 20\%$. For other cases (i.e., $b_\varepsilon > 20\%$), $\oplus 1$ is considered to achieve higher embedding capacity. For the rest of the discussions, the phrase 'proposed method' refers to the enhancement of RB combined with T1 or T2 as suggested by the aforementioned guideline.

3.6.2 Image Quality

Table 3.6 records the SSIM (Structural Similarity Index Measurement) (Z. Wang, Bovik, Sheikh, & Simoncelli, 2004) and PSNR (Pixel Signal-to-Noise Ratio) [dB] for the proposed method using the Caltech 101 dataset for various b_ε. In general, lower SSIM value (magnitude) and PSNR indicate more severe distortion. Experimental results suggest that the lowest and highest SSIM (PSNR) produced by the proposed method are 0.0850 (28.0438 dB) and 0.7866 (38.4184 dB), respectively. The proposed method

(a) Original (b) Output for b_ε = 100% (c) Output for b_ε = 50%

(d) Output for b_ε = 10% (e) Output for b_ε = 5% (f) Output for b_ε = 1%

Figure 3.10: Original and output images generated by the proposed method (without scrambling) for various b_ε

achieves more severe distortion when b_ε is small because larger amount of external data can be inserted into the image. The proposed method also demonstrates its capability in achieving progressive quality degradation by tuning the parameter b_ε because the SSIM and PSNR values decrease when b_ε decrease, and vice versa. Note that the PSNR for $b_\varepsilon = 100\%$ (i.e., 36.2568 dB) is lower than that for $b_\varepsilon = 90\%$ (i.e., 38.4184 dB) because most of the images in the dataset span the entire dynamic range, i.e., $r = 2^K$, for $b_\varepsilon = 100\%$. In other words, data embedding cannot be carried out because there is no available bins to be associated with the occupied bins. Hence, when these output images are compared to their original counterparts, the MSE (mean square errors) is zero, which produces infinity as the PSNR value. Here, the PSNR value of 36.2568 dB for $b_\varepsilon = 100\%$ is the average PSNR for the processed images only, i.e., those images that span the entire dynamic range are not included in the calculation of the average PSNR value.

For visual inspection purposes, **Figure 3.10**(a)$\sim$(f) show the original test image (i.e., Peppers) and its corresponding output generated by the proposed method (without scram-

(a) Original (b) Output for $b_\varepsilon = 100\%$ (c) Output for $b_\varepsilon = 50\%$

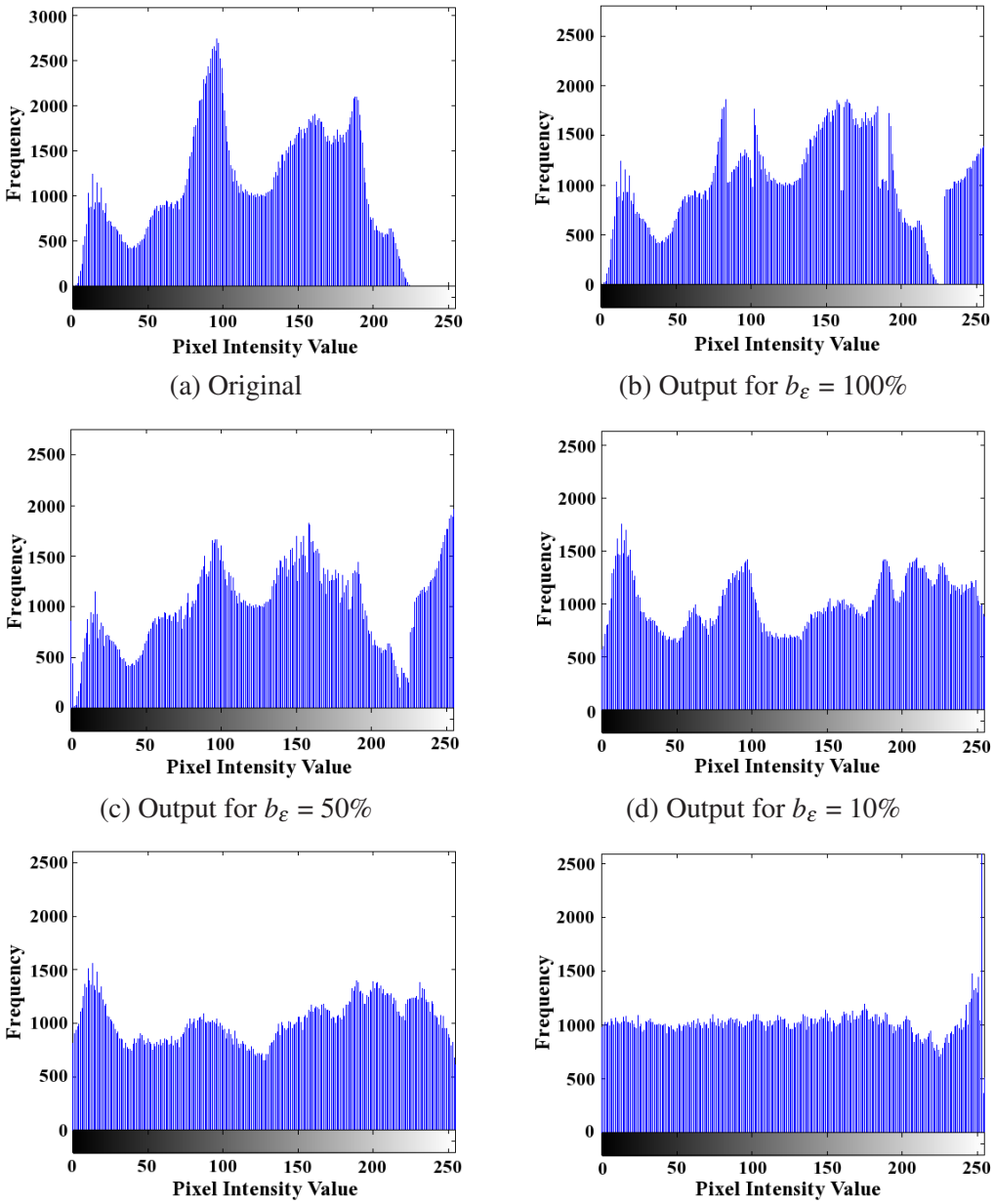
(d) Output for $b_\varepsilon = 10\%$ (e) Output for $b_\varepsilon = 5\%$ (f) Output for $b_\varepsilon = 1\%$

Figure 3.11: Original and output images generated by the proposed method (with scrambling) for various b_ε

bling). It is obvious that smaller b_ε generates more distortion, and vice versa. As expected, uneven distortion is observed in **Figure 3.10**(c), which supports the proposed idea of scrambling the output image within non-overlapping block of size $b_s \times b_s$ pixels to regulate the distortion. The corresponding embedded and scrambled images are shown in **Figure 3.11**(a)∼(f). It is observed that the distortion become more even after the scrambling process.

Figure 3.12(a)∼(f) show the distribution of pixel intensity values in the processed Peppers for various b_ε using the proposed method. It is observed that the histograms become more evenly distributed when b_ε decreases. In particular, when b_ε is 1% (i.e., approximately 5×5 pixels since the average image size in Caltech 101 dataset is 500×500 pixels), an almost flat histogram is attained, suggesting a near random image.

For completion of discussion, the graph of capacity versus distortion of the proposed method using the Caltech 101 dataset is plotted in **Figure 3.13**. As expected, distortion increases as the amount of data (i.e., gross embedding capacity) inserted increases.



(a) Original

(b) Output for $b_\varepsilon = 100\%$

(c) Output for $b_\varepsilon = 50\%$

(d) Output for $b_\varepsilon = 10\%$

(e) Output for $b_\varepsilon = 5\%$

(f) Output for $b_\varepsilon = 1\%$

Figure 3.12: Original and output histograms generated by the proposed method for various b_ε

However, higher distortion does not always imply higher achievable effective embedding capacity. This is because distortion is high when b_ε is small (e.g., 1%), but the side information involved consumes much of the gross embedding capacity, reducing the effective embedding capacity.

Results suggest that smaller b_ε (i.e., 1%) is not suitable for data embedding purposes because it offers low effective embedding capacity due to the need of storing large amount of side information. The empirical results suggest that the highest effective embedding

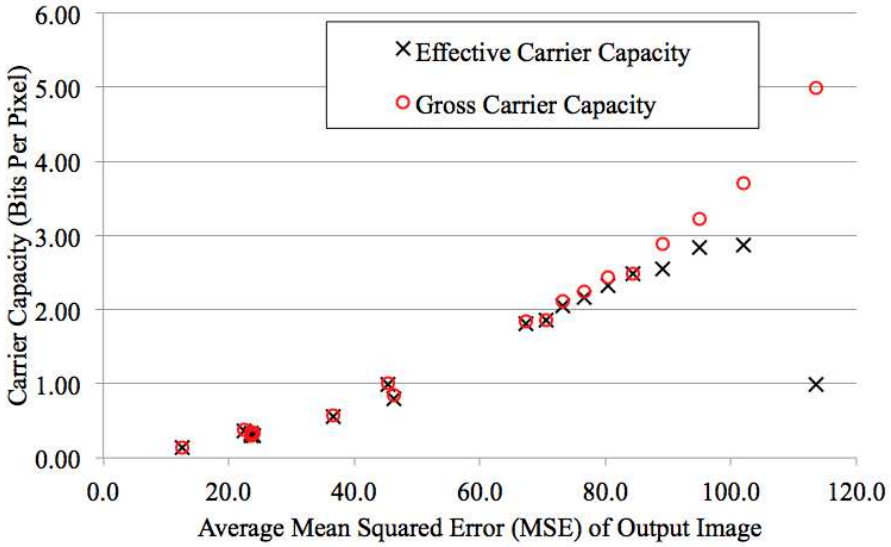


Figure 3.13: Capacity versus distortion graph for the proposed method

Table 3.7: Comparison of embedding capacity [bits per pixel] for related works and proposed method using Lenna

b_ε (pixels)	512×512	256×256	128×128	64×64	32×32	16×16
Ni et al. Method [1]	0.0105	0.0185	0.0265	0.0404	0.0588	0.0853
Huang et al. Method [2]			0.0733			
Proposed Method	0.3089	0.5053	0.6829	0.8935	1.4636	2.1252

[1] Ni et al. Method (Ni et al., 2006) without subtracting the venue utilized to store peak point (i.e., side information).
[2] Huang et al. Method (Huang & Fang, 2011) without subtracting the venue utilized the block map (i.e., side information). Threshold utilized for composition is 0.29 and within the block size of 16, 32, and 64.

capacity is achieved when $b_\varepsilon \sim 2\%$. On the other hand, it is observed that smaller b_ε can be utilized if the user intends to distort the host image severely (i.e., near random). Nevertheless, we emphasize that the desired level of distortion could be achieved by controlling b_ε (or in other words restricting the amount of information embeddable in the image) and b_s.

3.6.3 Comparison with Existing Methods

The proposed method is compared with related histogram-based data embedding methods, namely Ni et al.'s method (Ni et al., 2006) and Huang et al.'s method (Huang & Fang, 2011), in terms of embedding capacity. Ni et al. proposed the first histogram shifting method for external data insertion. For fair comparison purpose, we modify Ni et

Table 3.8: Comparison of SSIM and PSNR [dB] for related works and the proposed method using standard test images

Image Quality	SSIM			PSNR [dB]		
	Lowest	Highest	Range	Lowest	Highest	Range
Bit-Plane	0.0096	0.9968	0.9872	8.8995	51.1751	42.2756
UCPF-VH [1]	0.1269	0.9484	0.8215	9.6590	37.9716	28.3126
UCPF-D [2]	0.0222	0.7577	0.7355	11.2142	25.3358	14.1216
Proposed Method [3]	0.0402	0.8108	0.7706	11.5361	25.8292	14.2931

[1] UCPH-VH is from (Wong & Tanaka, 2010b).
[2] UCPH-D's (Wong & Tanaka, 2010a) experiment is conducted using 45 degree scrambling only.
[3] Lowest SSIM and PSNR are obtained using b_ε=2×2 pixels and the highest SSIM and PSNR are obtained using b_ε=512×512 pixels using $\oplus$2.

al.'s method by applying their original idea to smaller non-overlapping blocks (Huang & Fang, 2011). On the other hand, Huang et al.'s method is a recent data embedding method which divides the image using quadtree decomposition technique and embeds external data using histogram shifting. **Table 3.7** records the embedding capacities for (Ni et al., 2006), (Huang & Fang, 2011) and the proposed method. The results of using Lenna as the representative test image are recorded. It is observed that the embedding capacity increases when b_ε decreases for the proposed method and Ni et al.'s method, which is similar to the trend observed in **Table 3.6**. However, results indicate that the proposed method outperforms the two related methods in terms of embedding capacity for various b_ε. Nevertheless, we emphasize that the proposed method is able to perform, by far, better than (Ni et al., 2006) and (Huang & Fang, 2011) because our purpose here is to distort quality of the image while (Ni et al., 2006) and (Huang & Fang, 2011) aim to maintain the quality.

Comparison with related works in terms of progressive quality degradation are summarized in **Table 3.8**. Here, we consider the Bit-Plane approach (i.e., progressively applying XOR operations to increasing number of bit-planes in an image), and both Wong et al.'s method (Wong & Tanaka, 2010b, 2010a). In particular, the average lowest SSIM, average highest SSIM, SSIM range (i.e., the highest SSIM minus the lowest SSIM), aver-

age lowest PSNR, average highest PSNR and PSNR range (i.e., the highest PSNR minus the lowest PSNR) for six standard test images are recorded. For simplicity, scalability is quantified by the range of SSIM and PSNR achieved by each scrambling method.

Results indicate that Bit-Plane method has the widest range of SSIM and PSNR (i.e., most scalable), which ranges from 0.0098 to 0.9967 for SSIM and 8.8995 dB to 51.1751 dB for PSNR. However, the number of distortion levels achievable in the Bit-Plane method is limited to $K = 8$ since there are only K bit-planes in an image. Here, the highest SSIM and PSNR are achieved by applying XOR to the 8th Bit-Plane (i.e., LSB) while the lowest SSIM and PSNR are generated by applying the XOR operation to all K bit-planes. UCPF-VH (Wong & Tanaka, 2010b) and UCPF-D (Wong & Tanaka, 2010a) exhibit moderate scalability in comparison but offer more distortion levels (i.e., greater flexibility) as compared to the Bit-Plane method. Although the proposed method has comparable SSIM and PSNR ranges with UCPF-VH and UCPF-D, perceptual quality of its output can be preserved or distorted more severely by tuning the parameters b_ε and b_s. More importantly, perceptual quality degradation in the proposed method is achieved by means of embedding external data into the host, which realizes data embedding and perceptual quality degradation in a single framework. On the other hand, the existing methods considered (Huang & Fang, 2011; Wong & Tanaka, 2010b, 2010a; Ni et al., 2006) can only serve one of the two purposes, i.e., either data embedding or scrambling only. Therefore, the proposed method offers more features and greater flexibilities as compared to the existing methods considered.

3.7 Summary

A novel reversible UIH method following the E2S approach was proposed to purposely degrade the perceptual quality of the host image. Histogram Association Mapping method was proposed to eliminate the expensive pre-processing steps and the over-

flow / underflow problems in the conventional histogram shifting method while offering both scalable embedding capacity and progressive perceptual quality degradation features. HAM is divides the image into blocks, therefore, data can be still embedded even in the case all bins in the histogram are occupied. The proposed method is able to achieve higher effective embedding capacity and offers more features when compared to the existing methods. In terms of effective embedding capacity, the proposed method could embed up to $\sim$2.88 bpp in the best case scenario. At the same time, the proposed method achieved comparable range of distortion as compared with the existing scrambling methods but it offers greater flexibilities in controlling the image quality.

As future work, we shall focus on producing degraded output image of even distortion throughout the image without relying on scrambling function (as discussed in **Chapter 5**).

CHAPTER 4

SCALABLE UNIFIED REVERSIBLE INFORMATION HIDING (SURIH) IN DCT COMPRESSED DOMAIN

In this chapter, a novel unified information hiding method (using STE approach) called SURIH is proposed to achieve scalable quality degradation in scrambling and scalable embedding capacity in data embedding. AC coefficients are permuted without restricting the scope of permutation to a particular block or subband while DC coefficients are shuffled to progressively intensify distortion. An adaptive scanning order is proposed to suppress bitstream size increment due to shuffling of coefficients. The virtual queue decomposition is then proposed as the data representation scheme to embed external data where its embedding capacity can be scalably increased without causing bitstream size increment. All the processes in SURIH are completely reversible where the embedded data can be removed and the original content can be perfectly reconstructed from its processed counterpart. The basic performance of SURIH is verified through experiments using various standard test images and the CalTech 101 dataset. The proposed method is also compared with the existing scrambling methods, reversible data hiding methods, and their combinations.

Brief review of the JPEG compression standard that is relevant to this work is first presented in **Section 4.1.1**. Then, various JPEG scrambling techniques together with their shortcomings are reviewed in **Section 4.1.2**. Finally, discussion on various reversible data embedding techniques and their shortcomings are detailed in **Section 4.1.3**. Objectives of the proposed method are defined in **Section 4.1.5**.

The rest of the chapter is organized as follows. **Section 4.2** presents the proposed scalable unified reversible data hiding method. **Section 4.3** describes the decoding and

reconstruction processes, and **Section 4.4** highlights the features of SURIH. **Section 4.5** shows the experimental results of comparing our system to the existing scrambling methods, reversible data embedding methods, and their combinations. **Section 4.6** gives concluding remarks.

4.1 Introduction

This section presents an overview of JPEG compression standard. Next, related works of both scrambling and data embedding methods in JPEG are briefly reviewed. Based on the literature reviews, problems of the current methods and the objectives of this work are derived.

4.1.1 Overview of JPEG Compression Standard

The pixel values in the input image are first shifted by -128, and the processed image is partitioned into non-overlapping blocks, each of 8×8 pixels. Each block is transformed into the frequency domain by using DCT (discrete cosine transformation) and each DCT coefficient is then subject to quantization w.r.t. (with respect to) the QT (quantization table) scaled by the specified quality factor. DC coefficients are encoded using DPCM (differential pulse-code modulation) where only the difference between the current and the previously coded value is stored. On the other hand, AC coefficients are subject to zigzag ordering, and they are encoded in the form of ZRV (ZeroRun Value) pairs. For example, suppose the following sequence of AC coefficients are encountered:

$$-7,0,0,0,4,0,0,-2,0,1,\underbrace{0,\cdots,0}_{53\text{ zeros}}. \tag{4.1}$$

Then, the resulting ZRV pairs are

$$(0,-7),(3,4),(2,-2),(1,1). \tag{4.2}$$

Both DC and ZRV pairs are further entropy coded using Huffman coding.

To decode an JPEG encoded image G of dimension $8M \times 8N$ pixels, where $M, N \in \mathbb{N}$ ($\mathbb{N}$ denoting the set of natural numbers), the DCT coefficients are reconstructed using the following equation after entropy decoding:

$$\text{rec}_{u,v}(i,j) = C \times G_{u,v}(i,j) \times QT_{u,v} \tag{4.3}$$

Here, $G_{u,v}(i,j)$ denotes the (u,v)-th quantized DCT coefficients in the (i,j)-th block in G, $QT_{u,v}$ denotes the (u,v)-th entry in QT, C is a constant, $1 \leq u,v \leq 8$, $1 \leq i \leq M$ and $1 \leq j \leq N$. The reconstructed coefficients are inversely transformed to the spatial domain, and the process is repeated for each 8×8 coefficient block, followed by a level shift of $+128$ at the end.

4.1.2 Scrambling Technology in JPEG

There are several means to distort the perceptual quality of a JPEG compressed image by exploiting its underlying coding structure. Some of the representative scrambling techniques include randomizing coefficient sign (Shi & Bhargava, 1998), modifying zigzag scanning order (Kailasanathan & Naini, 2002), shuffling ZRV pairs within a block (Takayama, Tanaka, Yoneyama, & Nakajima, 2006), shuffling coefficients within the same subbands (W. Li & Yuan, 2007), permuting the variable length codewords (Bhargava, Shi, & Wang, 2004), mapping coefficients using bijective function (Takayama et al., 2008) and performing XOR operation on the appended bits in JPEG bitstream (Qian, Zhang, & Wang, 2014). However, Li et al. (W. Li & Yuan, 2007) pointed out that most of these aforementioned scrambling methods are not completely secure. In particular, an outline of the original JPEG image can be sketched directly from the scrambled image generated by (Takayama et al., 2008; Shi & Bhargava, 1998; Takayama et al., 2006), and

(Bhargava et al., 2004) by thresholding the number of nonzero coefficients in each block. Although Li et al.'s scrambling method (W. Li & Yuan, 2007) is secured w.r.t. their proposed sketching attack, their method suffers from severe bitstream size increment. Similarly, (Takayama et al., 2008) faces the same bitstream size increment problem although it offers the scalable scrambling functionality.

4.1.3 Reversible Data Embedding Method in JPEG

Reversible data hiding is commonly realized in JPEG compressed image by manipulating the AC coefficients (Ogihara, Nakamura, & Yokoya, 1996). One approach is to compress the LSB (Least Significant Bit) of a judiciously selected subband and embed this compressed data along with the external data (Fridrich et al., 2001). The relation between QT and the AC coefficients is also exploited to embed external data. Pre-defined groups of AC coefficients are also utilized for reversible data hiding purposes (C.-Y. Lin, Chang, & Wang, 2008). Diagonal bands of coefficients are first defined and nonzero coefficients are inserted at a location determined by the external data. Recently, unutilized Huffman codewords are identified and each of them is modified to encode a ZRV pair that is actually utilized in coding the image (Mobasseri et al., 2010). Each pair of unutilized-utilized codewords is exploited to encode data, where original and identified (to be unutilized) codewords can encode '0' and '1', respectively. Most of the aforementioned reversible data embedding methods suffer from generating an output with significant bitstream size increment, especially when more data is embedded into the image.

4.1.4 Problems of the Current Methods

The intrinsic problem of scrambling or data embedding in JPEG compressed image is the bitstream size increment. Therefore, for some scrambling methods, they restricted the scope of permutation to reduce the bitstream size increment. This will lead to the leakage of information as reported by (W. Li & Yuan, 2007).

4.1.5 Objectives

This chapter proposes a unified information hiding method in JPEG compressed domain following the STE approach, to realize scrambling-embedding while achieving scalability and reversibility. In addition, the proposed method permutes the coefficients within the image (i.e., broaden the scope of permutation) while embedding data into the coefficients. An adaptive scanning order is then proposed to suppress the bitstream size increment in the output image.

4.2 The proposed method: SURIH

In this section, a scalable unified reversible information hiding method is proposed in the JPEG compressed domain to achieve scalability in quality degradation for scrambling and scalability in embedding capacity for data embedding. The flow of processes is summarized in **Figure 4.1**. **Section 4.2.1** pre-processes the image, aiming to reduce bitstream size of the input image by modifying the scanning order. Data embedding and subtle scrambling are achieved in **Section 4.2.2**. Quality degradation is further intensified in **Section 4.2.3**.

4.2.1 Pre-processing

The input JPEG compressed image is pre-processed to reduce its bitstream size by means of reducing the distance (i.e., run of zeros) between two consecutive nonzero AC coefficients and maximizing the number of zeros coded by the End-of-Block codeword (Pennebaker & Mitchell, 1992). First, the number of nonzero coefficients in each of the 63 AC frequency subbands are captured, i.e.,

$$sb(u,v) = \sum_{i=1}^{M} \sum_{j=1}^{N} \delta_{u,v}(i,j), \tag{4.4}$$

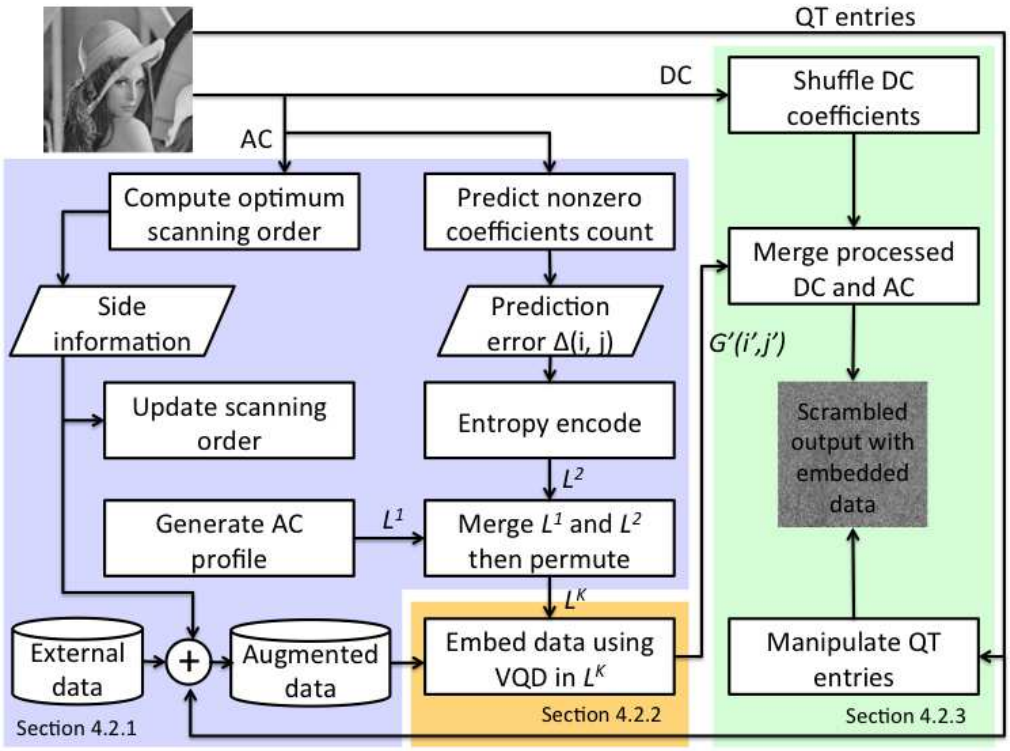


Figure 4.1: Flow chart for SURIH

where

$$\delta_{u,v}(i,j) = \begin{cases} 1 & \text{if } G_{u,v}(i,j) \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

With a change of 2D index to 1D subscript such as $w = (u-1) \times 8 + v$, $sb(u,v)$ can be linearized to $sb(w)$. Next, $sb(w)$'s are sorted in descending order so that $sb(w'_1) \geq sb(w'_2) \geq sb(w'_3) \geq ... \geq sb(w'_{63})$ and the new scanning order (denoted by SO_8) is constructed to visit the frequency subband represented by w'_1 first, followed by w'_2, w'_3, and so forth. **Figure 4.2** shows the default zigzag ordering SO_3 and the new scanning order SO_8 for Lenna compressed at QF of 80.

The static scanning orders proposed by Pan (Pan, 2002) (denoted as SO_6 and SO_7) and Itoh (Itoh, 2003) (denoted as SO_1, SO_2, SO_4, SO_5) are also adopted. Altogether, there are eight scanning choices to consider in each block, and the corresponding Huffman codewords w.r.t. each scanning order SO_{ind} are generated. Subscript *ind* of the scanning

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

(a) Default zigzag order

1	2	4	6	18	24	30	39
3	5	7	10	15	26	31	33
9	8	11	14	21	29	34	35
12	13	16	20	25	37	41	42
17	19	23	27	32	45	46	44
22	28	36	38	40	49	53	50
43	47	54	51	48	56	58	52
57	61	62	60	55	64	63	59

(b) Adaptive scanning order

Figure 4.2: Two scanning orders for Lenna

order SO_{ind} that produces the shortest complete Huffman codewords[1] is selected and output as side information. This process is repeated for all blocks and all selected SO_{ind}'s are combined with the original external data to form the augmented data, i.e., $[\{\mathrm{SO}_{ind}\}_{ind=1}^{M\times N}$, external data$]$. One may permute the selected SO_{ind}'s prior to the combining process to further complicate unauthorized reconstruction of the original image.

Each block $G(i,j)$ is finally linearized using the optimal scanning order as stipulated by the side information. Here, the total number of bits required to store the AC coefficients is expected to be smaller than that of the original image. The list of ZRV's in each block $G(i,j)$ is then output as the AC profile. As an example, the ZRV pairs in **Equation (4.2)** form the AC profile of the block in **Equation (4.1)**. AC profiles from all $G(i,j)$'s are concatenated in raster order to form the list of ZRV pairs L_1. An empty image $G^e_{u,v}(i,j) = 0$ of the same size as the original image is created, which becomes the output image eventually.

Next, the number of nonzero coefficients in $G(i,j)$, denoted by $\zeta(i,j)$, is stored for reversibility purposes. This information is needed to tell apart, how many ZRV pairs belong to each block in a sequential manner because all AC profiles are concatenated in

[1]Referring to the symbol (bit sequence) for zerorun, together with the actual coded value. For example, the ZRV pair $(1,6)$ is encoded as 1111001110, where the preceding 1111001 records the fact that a run of 1 zero for category 3 is encountered, and the value 6 is coded using the bit sequence of 110.

L_1. It is observed that ζ's are highly correlated, i.e.,

$$\zeta(i,j) \sim \zeta(i \pm e_i, j \pm e_j) \tag{4.6}$$

for small e_i and e_j. Therefore, we use predictive coding to store the number of nonzero coefficients in all blocks. In particular, $\zeta(i,j)$ is predicted to be $0.5 \times (\zeta(i,j-1) + \zeta(i-1,j))$, i.e., average of its west and north neighbors, because it is found to be the best prediction empirically. Regardless of the prediction mode, the distribution of errors Δ is expected to peak around zero and decay exponentially away from the center (i.e., resemble Laplace distribution). Thus, the errors $\Delta(i,j)$'s are further entropy coded. For simplicity, each Δ is mapped to a unique ZRV pair in JPEG. Specifically, the Huffman codewords extracted from the JPEG stream are sorted based on the complete Huffman codeword length. In the default table, (zerorun, value) = $(0,1)$ and $(0,-1)$ in category 1 are the shortest codewords, followed by $(0,-3),(0,-2),(0,2),(0,3)$ in category 2, and so forth. As a result, each of the possible 127 errors (since $-63 \leq \Delta(i,j) \leq 63$) is represented by a codeword in category $1,2,\cdots,6$. The more probable error is assigned to shorter codeword, and vice versa. To avoid coding the table explicitly, we map $\Delta = 0$ to $(0,-1), \Delta = -1$ to $(0,1), \Delta = 1$ to $(0,-3), \Delta = -2$ to $(0,-2)$, and so forth. Hence, one new ZRV pair is generated to capture the prediction error for each block.

These newly generated ZRV pairs are en-queued to a list L_2 in raster order. L_2 is concatenated to the end of L_1 (or combined by other means) to form L. Here, the list L contains all ZRV pairs extracted from G and the additional $M \times N$ ZRV pairs introduced to record $\zeta(i,j)$. L undergoes a permutation using the secret key κ to generate the permuted list L^κ. Finally, ZRV pairs in L^κ are evenly distributed into groups of n blocks (i.e., $M \times N/n$ groups in total) to encode external data while generating the output image as described in **Section 4.2.2**.

4.2.2 Data Embedding Using Virtual Queue Decomposition

In this section, VQD (virtual queue decomposition) is proposed to embed data. VQD has three advantages over the existing data representation schemes, namely: (a) it is reversible; (b) it does not require side information such as the location map to decode the embedded data (Tian, 2003; Kawaguchi & Eason, 1998; Coltuc, 2011); (c) it can increase embedding capacity without causing bitstream size increment. VQD is applied on the permuted list L^{κ} to embed the augmented data.

Let $\{x_h\}_{h=1}^{t}$ be a sequence (i.e., queue) of t ZRV pairs, where ZRV pairs are of the form shown in **Equation (4.2)**. VQD splits $\{x_h\}_{h=1}^{t}$ into n sub-queues and it allows a sub-queue to be empty (i.e., no ZRV pairs). The first sub-queue is labeled as s_1, second sub-queue as s_2, and so forth. Here, the relative order among ZRV pairs within any s_i for $1 \leq i \leq n$ is maintained and prohibited from changing. Let $|s_i|$ denote the cardinality (i.e., number of ZRV pairs) of sub-queue s_i. $\{x_h\}_{h=1}^{t}$ is split into sub-queues that produce a certain combination of n-tuple $(|s_1|, |s_2|, \cdots, |s_n|)$ to encode the external data. For example, $s_1 = \{\}, s_2 = \{x_1\}$ and $s_3 = \{x_2, x_3, x_4\}$ is a specific splitting of $t = 4$ ZRV pairs into $n = 3$ sub-queues, generating the 3-tuple of $(0, 1, 3)$. Note that there is no change in the length of zerorun for each ZRV pair and the original sequence $\{x_h\}_{h=1}^{t}$ is readily reconstructed when the sub-queues s_i are sequentially concatenated.

As an illustration, consider a queue of $t = 4$ ZRV pairs to be decomposed into $n = 3$ sub-queues. **Table 4.1** lists all possible ways of splitting a queue of 4 ZRV pairs into 3 sub-queues. Since there are 15 combinations, $\lfloor \log_2(15) \rfloor = 3$ bits of the external data can be encoded by using $(|s_1|, |s_2|, |s_3|)$. Specifically, the combination of $(0, 0, 4)$ can be assigned to encode the three-bit external data of '000', $(0, 1, 3)$ to '001', and so forth as shown. These assignments can also be permuted in a periodic or non-periodic manner to complicate unauthorized decoding of the embedded data.

Table 4.1: List of all theoretically possible decompositions for queue of 4 items in 3 sub-queues

Case	s_1	s_2	s_3	$(\|s_1\|,\|s_2\|,\|s_3\|)$	Ext. Info
1	-	-	x_1,x_2,x_3,x_4	$(0,0,4)$	000
2	-	x_1	x_2,x_3,x_4	$(0,1,3)$	001
3	-	x_1,x_2	x_3,x_4	$(0,2,2)$	010
4	-	x_1,x_2,x_3	x_4	$(0,3,1)$	011
5	-	x_1,x_2,x_3,x_4	-	$(0,4,0)$	100
6	x_1	-	x_2,x_3,x_4	$(1,0,3)$	101
7	x_1	x_2	x_3,x_4	$(1,1,2)$	110
8	x_1	x_2,x_3	x_4	$(1,2,1)$	111
9	x_1	x_2,x_3,x_4	-	$(1,3,0)$	Not used
10	x_1,x_2	-	x_3,x_4	$(2,0,2)$	Not used
11	x_1,x_2	x_3	x_4	$(2,1,1)$	Not used
12	x_1,x_2	x_3,x_4	-	$(2,2,0)$	Not used
13	x_1,x_2,x_3	-	x_4	$(3,0,1)$	Not used
14	x_1,x_2,x_3	x_4	-	$(3,1,0)$	Not used
15	x_1,x_2,x_3,x_4	-	-	$(4,0,0)$	Not used

To apply VQD in an JPEG compressed image, every n blocks $G^e(i,j)$'s are grouped together and a block within each group plays the role of a sub-queue. In total, there are $(M \times N)/n$ groups altogether. Each group of n blocks is assigned to host $t = n \times \bar{\zeta}$ ZRV pairs where

$$\bar{\zeta} = \frac{|L_2|}{M \times N} + \frac{|L_1|}{M \times N} = 1 + \frac{1}{M \times N} \sum_{i}^{M} \sum_{j}^{N} \zeta(i,j). \tag{4.7}$$

In particular, the first $t = n \times \bar{\zeta}$ ZRV pairs are de-queued from L^{κ} and assigned to the first group, the next t ZRV pairs are assigned to the second group, and so forth. The data embedding process then takes place by applying VQD within each group. Specifically, the table of combinations (similar to **Table 4.1**) is referred to select the corresponding combination $(|s_1|,|s_2|,\cdots,|s_n|)$ for storing (encoding or representing) segment of the external data. In other words, the external data dictates which combination to utilize. Once the combination is decided, the first $|s_1|$ ZRV pair(s) for this group become element(s) of the first block, the following $|s_2|$ ZRV pair(s) become element(s) of the second block, and so forth. For example, suppose $n = 3, t = 4$ and the external data segment to be embedded is '001'. VQD refers to the last column of **Table 4.1** and decides that the second

combination is to be imposed. Therefore, no AC coefficient is assigned to the first block, the first AC coefficient is assigned to the second block, and the remaining (three) coefficients that follow are assigned to the third block. This process is repeated for all groups of n blocks. Note that the table of combinations can be easily regenerated as described in **Section 4.2.2** when n and t are given. It should be noted that the encoding process at this stage only updates the AC coefficients in $G^e(i,j)$ without updating the DC coefficients (i.e., all DC values remain 0's).

Since an 8×8 block can accommodate at most 63 AC coefficients, some of the combinations are practically not viable. For example, if $zr(x_3) + zr(x_4) + 2 > 63$ where $zr(x_h)$ denotes the length of zeros in x_h, then x_3 and x_4 cannot be put in the same block (sub-queue). In that case, only cases $4, 8, 11, 13$ and 14 underlined in **Table 4.1** can be utilized, reducing the embedding capacity from 3 bits to $\lfloor \log_2(5) \rfloor = 2$ bits. Note that **Table 4.1** is always referred to for the case of $t = 4$ ZRV and $n = 3$ sub-queues, but the underlined viable cases depend on the zerorun-length of the ZRV pairs under consideration.

The newly updated blocks $G^e(i,j)$ (without the DC value) are output as $G'(i,j)$. To avoid unauthorized decoding of the embedded data, $G'(i,j)$ are shuffled to $G'(i',j')$, i.e., similar to the jigsaw puzzle. We shall show in **Section 4.4** that n can be increased to improve embedding capacity, i.e., scalability in embedding capacity.

4.2.3 Scalable Scrambling

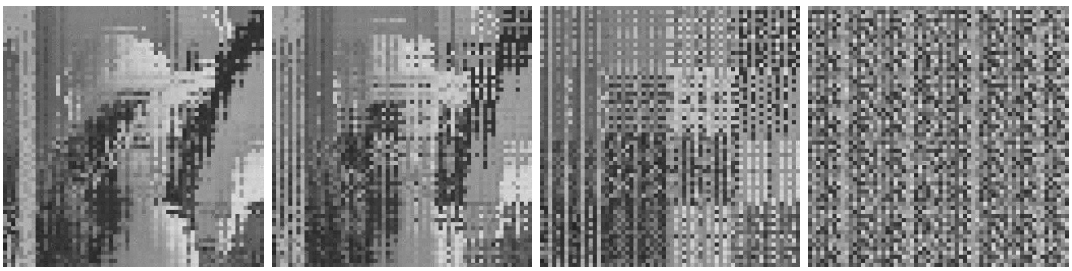
Here, we describe two methods to control image quality degradation after pre-processing and data embedding. **Figure 4.3(a)** shows the image of Lenna after undergoing the processes described in **Section 4.2.1** and **Section 4.2.2**, while keeping the original DC coefficients intact only for display purposes. It can be seen that awkward checkerboard-like patterns are visible throughout the processed image because the AC coefficients are permuted. These patterns can be suppressed to achieve higher perceptual quality or intensi-

(a) Original QT (b) QT = 1 (c) QT = 255

Figure 4.3: Scrambled image with embedded data

fied to cover the outline of the image by exploiting QT in **Equation (4.3)**. Specifically, magnitude of the entries in QT can be decreased to suppress the checkerboard-like patterns because the reconstructed AC coefficients become smaller and the contribution of each DCT basis pattern is reduced. Magnitude of the entries in QT can also be increased to intensify distortion because the reconstructed coefficients are saturated to larger magnitudes. **Figure 4.3(b)** and **(c)** show the resultant images by changing entries in QT of **Figure 4.3(a)** to unity and 255, respectively. It clearly shows that the perceptual quality of the image is improved or degraded by controlling the entries in QT. It should be noted that changing the QT entries to any value in the range of $[1,255]$ does not affect the JPEG bitstream size (Pennebaker & Mitchell, 1992). SURIH can afford this change because it can store the original QT entries employed during compression as part of the augmented data (i.e., $[\{\mathrm{SO}_{ind}\}_{ind=1}^{M\times N}$, QT, external data]) to ensure complete reconstruction of the original image. However, the general outline of the image is still visible, even when it is covered by bizarre checkerboard-like patterns shown in **Figure 4.3(c)**. Furthermore, a sketch of the original image can be easily revealed by using the intact DC components.

To further increase the distortion level, we propose another method to manipulate the DC coefficients. To this end, all DC coefficients are extracted to form a matrix of dimension $M \times N$. This matrix is then divided into non-overlapping $W \times W$ windows ($1 \leq W \leq \min\{M,N\}$). Finally, DC coefficients in each window are shuffled using a



(a) $W = 4$ (b) $W = 8$ (c) $W = 16$ (d) $W = 64$

Figure 4.4: Example of scrambled images with various values of W

secret-key-based random sequence and the shuffled DC coefficients are put at their respective locations to degrade the perceptual quality. Here, W controls the distortion level, where the smallest W (i.e., $W = 1$) and the largest W (i.e., $W = \min\{M,N\}$) produce the least and the most distorted output image, respectively. **Figure 4.4** shows the results of shuffling DC coefficients of **Figure 4.3(b)** using various window sizes of $W = 4, 8, 16$, and 64. It clearly shows that the distortion level is intensified as W increases. Scalable quality degradation can be effectively achieved by manipulating the entries in QT and shuffling the DC coefficients within various window sizes. For example, limited detail of the content shown in **Figure 4.4(a)** can be considered in online photo store to attract potential buyers. The degraded noise-like content shown in **Figure 4.4(d)** can be utilized in document management where the inferior officer handles all confidential documents without knowing the visual content. A completely unintelligible content can be created by setting all entries in QT as 255 and shuffling the DC coefficients in each of the $W_{\max} \times W_{\max}$ windows, where $W_{\max} = \min\{M,N\}$.

Finally, the manipulated DC coefficients, AC coefficients, and QT entries are consolidated to generate the output image. Modified QT entries are coded in the header section of the output JPEG image G'. The manipulated blocks $G'(i', j')$ (processed by VQD) and the shuffled DC components are put together at their respective locations to form G', which is now scrambled (partially or entirely depending on the application) and embedded with external data.

4.3 Decoding Embedded Data and Reconstruction of Original Image

The value n utilized during data embedding is required to extract the embedded data. When the blocks $G'(i', j')$ are de-shuffled to $G'(i, j)$, groups of n blocks are re-formed. Within a group, the cardinality of each block (i.e., the number of ZRV pairs) is computed to form the n-tuple $(|s_1|, |s_2|, \cdots, |s_n|)$. The same table considered during data embedding can be regenerated because the total number of ZRV pairs in a group of n blocks remains unchanged regardless of the cardinality of each block (i.e., regardless of the splitting process). In other words, $t = \sum_i^n |s_i|$ can be directly computed from the n blocks in $G'(i, j)$. Nonviable combinations (i.e., combinations that assign ZRV pairs to more than 63 locations in a block) can also be verified because the relative order among ZRV pairs are kept intact. The remaining viable combinations are therefore utilized to decode the embedded data. The bit sequence associated with the n-tuple is the output. Using the previous example, if the triplet $(0, 1, 3)$ is encountered, then '001' is output. The process is repeated for all groups, and the output bit sequences are concatenated to form the (decoded) augmented data.

To completely reconstruct the original image, profile of each block $G'(i, j)$ is extracted and merged to obtain the list L^{κ}. L is reconstructed after de-shuffling L^{κ} with the key κ. Then, L_2 is derived from the last $M \times N$ ZRV pairs in L and utilized to guide the redistribution process of ZRV pairs from L_1 into each respective 8×8 coefficient block. The original scanning order in each block is obtained by referring to $\{\mathrm{SO}_{ind}\}_{ind=1}^{M \times N}$ captured in the augmented data. De-shuffling the DC coefficients completes the reconstruction process.

4.4 Features in SURIH

SURIH differs from simply merging a reversible data embedding method (e.g., (Takayama et al., 2008)) and a scalable scrambling method (e.g., (Xuan et al., 2009)). In SURIH, the

embedded data could be decoded directly from the processed image G' without revealing the original image, i.e., the STE approach. To achieve the same feature in the merged approach, the external data must be embedded into the scrambled image (i.e., first scramble then embed), otherwise the original image G must be reconstructed prior to decoding the embedded external data. SURIH also differs from the prior art because it offers scalable functionalities in terms of quality degradation and embedding capacity.

The following subsections further highlight the features in SURIH: **Section 4.4.1** discusses VQD in detail, **Section 4.4.2** elaborates on the scope of permutation and robustness against unauthorized viewing, and finally **Section 4.4.3** describes bitstream size increment in SURIH and its suppression.

4.4.1 Scalable Carrier Capacity in VQD

In this section, we explain how scalable embedding capacity is achieved in VQD. Given a sequence of t ZRV pairs $\{x_h\}_h^t$, we consider its number of theoretically possible decompositions into n sub-queues and denote that number by $\tau(\mathfrak{n},\mathfrak{t})$. Mathematically, $\tau(n,t)$ is defined as

$$\tau(n,t) = \left| \left\{ (|s_1|,|s_2|,\cdots,|s_n|) : t = \sum_{i=1}^{n} |s_i| \right\} \right| \tag{4.8}$$

while preserving the relative order of each ZRV pair. **Table 4.2** records $\tau(n,t)$ for $n = 1,2,\cdots,8$ sub-queues and $t = 1,2,\cdots,12$ ZRV pairs. It is observed that $\tau(n,t)$ increases as n (i.e., in horizontal direction) or t (i.e., in vertical direction) increases. Since an image has a fixed number of ZRV pairs, we shall increase the embedding capacity of SURIH by means of increasing n while enforcing $t = \bar{\zeta} \times n$.

As an illustration of the improvement in embedding capacity by increasing n, let us refer to **Table 4.2** and consider the scenario of 12 ZRV pairs with different values of n

Table 4.2: The Number of Theoretically Possible Combinations $\tau(n,t)$ for Various n and t

n	1	2	3	4	5	6	7	8
$t=1$	1	2	3	4	5	6	7	8
$t=2$	1	3	6	10	15	21	28	36
$t=3$	1	4	10	20	35	56	84	120
$t=4$	1	5	15	35	70	126	210	330
$t=5$	1	6	21	56	126	252	462	792
$t=6$	1	7	28	84	210	462	924	1716
$t=7$	1	8	36	120	330	792	1716	3432
$t=8$	1	9	45	165	495	1287	3003	6435
$t=9$	1	10	55	220	715	2002	5005	11440
$t=10$	1	11	66	286	1001	3003	8008	19448
$t=11$	1	12	78	364	1365	4368	12376	31824
$t=12$	1	13	91	455	1820	6188	18564	50388

such as $n = 2, 3$ and 6. Assuming all theoretically possible decompositions are also practically achievable (i.e., $zr(s_1)+zr(s_2)+\cdots+zr(s_{12})+12 \leq 63$), $3 \times \lfloor \log_2(\tau(2,4)) \rfloor = 3 \times \lfloor \log_2(5) \rfloor = 6$ bits are attained by considering three groups of 2 sub-queues, with 4 ZRV pairs in each sub-queue. In case of two groups of 3 sub-queues, $2 \times \lfloor \log_2(\tau(3,6)) \rfloor = 2 \times \lfloor \log_2(28) \rfloor = 8$ bits are achieved. Finally, $\lfloor \log_2(\tau(6,12)) \rfloor = \lfloor \log_2(6188) \rfloor = 12$ bits are attained by considering one group of 6 sub-queues. Similar argument applies to other cases, and hence we expect embedding capacity to scale according to n. This claim is verified in **Section 4.5**. Interestingly, the values recorded in **Table 4.2** are observed to be symmetric w.r.t. the super diagonal (underlined). These values also satisfy the following recurrence equation, which is proved in **Appendix A**:

$$\tau(n,t) = \tau(n-1,t) + \tau(n,t-1). \tag{4.9}$$

We implemented n-nested *for*-loops using multi-threading to search for n-tuples $(|s_1|, |s_2|, \cdots, |s_n|)$ that satisfy $|s_1| + |s_2| + \cdots + |s_n| = t$. Such tuples are en-queued to a common list accessible by all threads. This populated common list is sorted using the first tuple as the primary key, the second tuple as the secondary key, and so forth. This

sorting process always rearranges these entries into the same order. For example, the list of all combinations for $(n,t) = (3,4)$ always appears in the order shown in **Table 4.1**. Thus, higher embedding capacity comes at the expense of higher computational complexity. Theoretically, one may increase n to the upper bound of $M \times N$, but it is not computationally viable for the actual application. Therefore, there is a trade-off between computational complexity and embedding capacity, but this has no impact to the bitstream size.

4.4.2 Scope of Permutation and Robustness against Unauthorized Decoding

The scope of permutation in the existing scrambling methods is limited to block level (i.e., nonzero coefficients or ZRV pairs are shuffled within a block) or subband level (i.e., coefficients from the same subband are shuffled). By design, SURIH permutes the coefficients to any subband in any block. Hence, SURIH is robust against nonzero coefficient count attack (W. Li & Yuan, 2007) as well as multiple sketch attacks (Minemura, Moayed, Wong, Qi, & Tanaka, 2012). Although it is a known fact that scrambling is achieved at the expense of sacrificing security (W. Zeng & Lei, 2003), it is worth considering the number of trials in the brute force attack (i.e., reconstruction of the original image by exhaustive trials) in SURIH. An attacker needs to consider

$$\underbrace{(M \times N)!}_{\text{DC component}} \times \underbrace{(|L|)!}_{\text{AC component}} \tag{4.10}$$

trials of permutation, and for each guessed permutation, the attacker still has to decide how many ZRV pairs should go into each block $G(i,j)$, which further complicates the attack. In addition, without giving any impact to the bitstream size, we can randomize the sign of each coefficient and map the coefficient to other values belonging to the same category (e.g., $-15,-14,\cdots,-8,8,\cdots,14,15$ in category 4).

Next, SURIH prevents unauthorized decoding of the embedded data by permuting the blocks $G'(i,j)$ to $G'(i',j')$. Permuting the blocks will only affect the output bitstream size insignificantly due to byte-alignment (Pennebaker & Mitchell, 1992). In order to correctly guess the embedded data, an attacker will need to attain the right order for the blocks (i.e., $(M \times N)!$ trials) and the value of n. Furthermore, the table of combination (similar to **Table 4.1**) could be shuffled to further complicate unauthorized decoding of the embedded data.

Therefore, it requires a significant number of trials in order to decode the embedded data and completely reconstruct the original image without the key κ and parameter n.

4.4.3 Bitstream Size Increment and Its Suppression

This section rationalizes that subtle bitstream size increment is acceptable and describes how this increment is suppressed in SURIH. The existing scrambling methods produce output content of larger bitstream size because the original zerorun-length property is destroyed (W. Li & Yuan, 2007) or the magnitude of coefficient is changed (Takayama et al., 2008; C. Wang et al., 2003). On the other hand, the existing reversible data embedding methods also cause bitstream size increment due to magnitude increment (Xuan et al., 2009), introduction of nonzero coefficients (C.-Y. Lin et al., 2008; Fridrich et al., 2001), or encoding the same ZRV pair using a longer Huffman codeword (Mobasseri et al., 2010). Therefore, it is fair to conclude that bitstream size increment in the output product is an intrinsic property of information hiding.

In SURIH, the output image is expected to assume larger bitstream size than its original counterpart because $M \times N$ foreign ZRV pairs are added to encode $\zeta(i,j)$. However, we emphasize that these ZRV pairs are introduced to allow shuffling without restriction to the block or subband level. The manipulation on DC coefficients also contributes to the bitstream size increment. When DC coefficients are shuffled, the efficiency of DPCM

decreases, resulting in an output of larger bitstream size. However, it is the permutation of DC components and the manipulation of entries in QT that allow SURIH to control the perceptual fidelity, and therefore achieve scalability in quality degradation. Note that the shuffling of L to L^{κ} does not affect the bitstream size since SURIH handles nonzero DCT coefficients directly in the form of ZRV pairs. Nevertheless, small fluctuation in bitstream size is expected due to byte-alignment (Pennebaker & Mitchell, 1992).

To compensate the inevitable increment of the bitstream size, the image is pre-processed by changing its scanning order as detailed in **Section 4.2.1**. The more scanning orders considered in pre-processing, the greater the suppression in bitstream size. In this study, we consider $SO_{\max} = 8$ scanning orders. **Figure 4.5** shows the frequency of each scanning order SO_{ind} chosen for various test images compressed at QF= 80 for $ind = 1, 2, 3, \cdots, 8$. It is obvious that the constructed scanning order SO_8 is selected most of the time ($\sim 40\%$), followed by the default zigzag scanning order SO_3. This shows the superiority of the proposed scanning order over the existing (static) ones in reducing length of zerorun. With $SO_{\max} = 8$, a gain of $\sim 4.5\%$ in compression w.r.t. JPEG is achieved, where this available room is then utilized to accommodate the newly added ZRV pairs. However, this suppression is achieved at the expense of storing the side information (i.e., the chosen *ind* value for each block), which reduces the effective embedding capacity of SURIH. Hence, there is a trade-off between the bitstream size suppression and the effective embedding capacity.

Last but not least, we emphasize that VQD does not further increase the bitstream size of the output image regardless of the value of n. The ZRV pairs are merely put into groups of n sub-queues without additionally introducing new ZRV pairs, changing the zerorun length, or promoting a coefficient to another category.

In summary, the number of scanning orders, bitstream size, and effective embedding capacity are interrelated. The more scanning orders one considers, the smaller the output

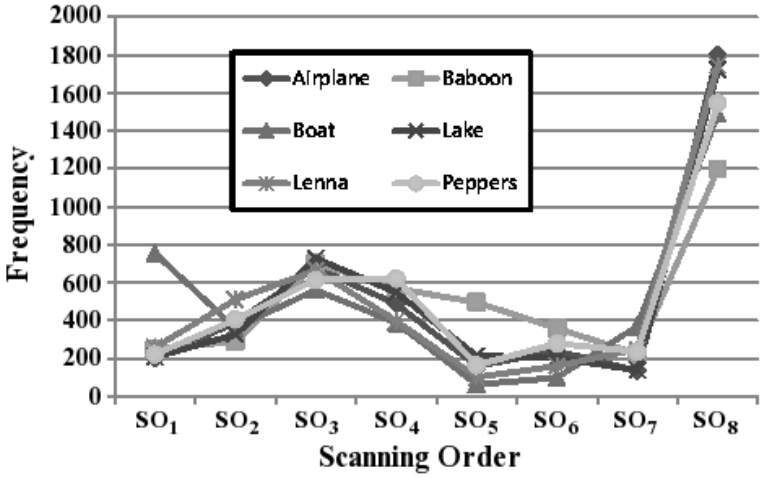


Figure 4.5: Statistics of selected scanning order

bitstream size. However, considering more scanning orders will reduce the effective embedding capacity because more bits are spent in coding *ind* in each block. Nevertheless, effective embedding capacity can be increased by increasing n without further causing bitstream size increment as discussed in **Section 4.4.1**.

4.5 Experimental Results

To verify the performance of SURIH, six 8-bit 512×512 standard grayscale test images, namely, Airplane, Baboon, Boat, Lake, Lenna and Peppers (*The USC-SIPI Image Database [online]*, n.d.), are considered. Unless specified otherwise, the quality factor QF is set at 80 and the QT entries are set to unity except for the DC component. All six images are scrambled while embedding a random bit sequence (i.e., external data) into them. By visual inspection, it is verified that the images could be distorted after the manipulation by SURIH, where distortion of the images is further intensified as W increases. Using the correct secret key(s), the embedded data could be extracted from the output of SURIH, and the original image can also be completely reconstructed. Last but not least, it is verified that nonzero coefficient count attack (W. Li & Yuan, 2007) failed to produce a sketch of the image processed by SURIH regardless of the imposed level of distortion.

4.5.1 Basic Performance of SURIH

Firstly, the gross embedding capacity (in raw bit count) $\Omega(G,n)$ achieved by VQD is recorded in **Table 4.3** for $n = 2,3,4,5$ and 6. For reference purposes, the total number of AC coefficients $|L|$ is also recorded. It is observed that, regardless of the image, embedding capacity increases as n increases, where the capacity of $n = 6$ is almost double that of $n = 2$. In addition, the embedding capacity for images of smaller $|L|$ (e.g., Airplane or Lenna) tends to scale better with n when compared to the embedding capacity for images of larger $|L|$ (e.g., Baboon). Nevertheless, statistics of ZRV pairs (i.e., zerorun-length) in an image also influences the embedding capacity w.r.t. VQD. Since part of the gross embedding capacity is utilized to store side information, the effective embedding capacity is computed as $\Omega(G,n) - M \times N \times \lceil \log_2(SO_{\max}) \rceil$, where $SO_{\max}$ denotes the number of scanning orders considered and $M \times N = 64 \times 64$ is the number of blocks in each test image. For example, if $SO_{\max} = 4$ (i.e., only 4 scanning orders are considered), the effective embedding capacity is $\Omega(G,n) - 8192$. In the case of $SO_{\max} = 1$, the gross and effective embedding capacities are the same.[2] It should be noted that using different $SO_{\max}$ results in small fluctuation in $\Omega(G,n)$, but it has significant impact on the effective embedding capacity. Nevertheless, the results recorded in **Table 4.3** are the average of the gross embedding capacity achieved by using $SO_{\max} = 1,2,4$ and 8.[3]

Secondly, the visual quality of the processed image is quantified by using SSIM (Z. Wang et al., 2004). Note that we are comparing the processed image g' to the original JPEG compressed image g (decoded from G' and G, respectively, to the spatial domain) instead of the raw (uncompressed) image. The value of SSIM $= 1$ is only attained when

[2] To improve coding efficiency for the side information, one may exploit the statistics of the chosen *ind* (e.g., **Figure 4.5**) and consider entropy coding (e.g., Huffman). In case of $SO_{\max} = 8$, experimental results suggest that ~ 2.3 bits per block are spent on coding the side information when using Huffman coding (versus 3 bits without entropy coding).

[3] Here, $SO_{\max} = 1$ denotes the case when only the zigzag scanning order is considered, $SO_{\max} = 2$ denotes the case where the two most contributive (among 8 of them) scanning orders (in terms of suppressing bitstream size increment) are considered, etc.

Table 4.3: Gross embedding capacity $\Omega(G,n)$ [bits] for various n

n	Airplane	Baboon	Boat	Lake	Lenna	Peppers
2	8609	6681	9357	9565	8649	8771
3	12118	8942	12634	12857	11947	11989
4	13984	10309	14522	14793	13808	13876
5	15154	11226	15695	15974	14970	15029
6	16015	12354	16516	16809	15807	15867
$\|L\|$	52532	111608	66087	71280	51382	53876

Table 4.4: SSIM ($\times 10^{-3}$) and PSNR (dB) for the images manipulated by SURIH

W	Airplane	Baboon	Boat	Lake	Lenna	Peppers
1	578 (21.6)	272 (19.5)	468 (21.9)	468 (20.6)	588 (23.5)	589 (22.7)
2	581 (18.2)	273 (18.2)	469 (18.6)	472 (16.8)	586 (19.2)	588 (18.2)
4	435 (16.6)	178 (17.3)	320 (17.1)	274 (14.3)	398 (17.4)	370 (16.0)
8	392 (14.9)	149 (16.1)	274 (15.8)	239 (13.3)	305 (15.0)	260 (13.1)
16	321 (13.7)	129 (15.0)	243 (14.6)	195 (12.6)	196 (13.2)	169 (11.9)
32	226 (12.7)	102 (13.7)	180 (13.2)	084 (9.6)	177 12.6)	120 (10.9)
64	195 (12.2)	104 (13.7)	156 (12.2)	077 (9.0)	130 (11.8)	115 (10.9)
Max	007 (5.9)	004 (6.4)	006 (6.4)	005 (5.8)	006 (6.5)	007 (6.3)
Range	574 (15.6)	269 (13.1)	463 (15.5)	467 (14.8)	583 (17.0)	582 (16.4)

$g = g'$, i.e., all the pixel values match. The representative SSIM results of using $W = 1,2,\cdots,64, n = 2$ and $SO_{\max} = 1$ are recorded in **Table 4.4**. The corresponding PSNR (peak signal-to-noise ratio) value is also recorded in parenthesis for reference purposes. Here, $W = 1$ denotes the case where only the AC coefficients are processed while DC coefficients remain intact at their original positions. Results suggest that the quality of the input image is distorted as soon as AC coefficients are scrambled because the recorded SSIM values for $W = 1$ are all less than unity. For each W, the distortion level attained differs from image to image. Distortion is further intensified when larger W is considered, verifying that scalability in quality degradation is achieved. For completion of discussion, the results for MAX are collected by setting all QT entries to 255 and using $W = 64$. Since the SSIM values for MAX are lower than those for $W = 64$, it verifies that distortion can be further intensified by manipulating the entries of QT. Similar arguments are applicable to the observed PSNR values.

Next, we consider the bitstream size increment (in percentage) after scrambling and

Table 4.5: Percentage of Bitstream Size Increment [%] after Scrambling and Data Embedding for Various W and $SO_{\max}$ [%]

	$SO_{\max}$	$W=1$	2	4	8	16	32	64	Max
Airplane	1	5.90	5.89	7.15	7.73	8.42	9.89	10.15	10.17
	2	3.89	4.76	5.17	5.76	6.54	7.99	8.18	8.18
	4	2.69	3.53	3.96	4.56	5.30	6.76	6.96	9.06
	8	1.61	2.43	2.85	3.47	4.17	5.65	5.86	5.86
Baboon	1	2.48	2.47	2.92	3.28	3.54	4.07	3.82	3.75
	2	1.21	1.43	1.64	2.03	2.27	2.84	2.54	2.51
	4	0.33	0.55	0.77	1.19	1.39	1.96	1.68	1.72
	8	-0.37	-0.16	0.07	0.46	0.71	1.26	0.99	0.97
Boat	1	4.92	4.90	5.87	6.16	6.69	7.07	7.61	7.58
	2	0.74	1.37	1.60	1.99	2.46	2.81	3.37	3.37
	4	-0.64	0.01	0.23	0.62	1.08	1.45	2.00	2.00
	8	-1.37	-0.73	-0.48	-0.10	0.33	0.74	1.23	1.27
Lake	1	4.54	4.57	5.68	5.95	6.59	8.54	7.83	7.87
	2	2.74	3.18	3.82	4.10	4.74	6.72	6.04	6.01
	4	1.69	2.22	2.82	3.09	3.75	5.74	5.04	5.01
	8	1.10	1.63	2.18	2.53	3.14	5.14	4.40	4.42
Lenna	1	5.95	6.00	7.08	8.27	9.94	9.93	10.43	10.43
	2	3.53	3.94	4.68	5.81	7.40	7.45	7.96	7.89
	4	2.33	2.81	3.52	4.71	6.26	6.32	6.79	6.78
	8	1.47	1.91	2.62	3.81	5.36	5.40	5.93	5.95
Peppers	1	6.01	6.00	7.44	8.60	9.62	10.45	10.49	10.50
	2	4.23	4.94	5.61	6.77	7.86	8.69	8.66	8.74
	4	2.66	3.40	4.05	5.23	6.26	7.11	7.13	7.12
	8	1.49	2.23	2.91	4.10	5.11	5.98	5.96	5.96

data embedding. The results are recorded in **Table 4.5** for $n = 2$ as the representative case. From the table, we observed that the image processed by SURIH is of larger bitstream size compared to its original counterpart when only the standard zigzag scanning order SO_3 (i.e., $SO_{\max} = 1$) is employed. This is an expected outcome because we introduce foreign ZRV pairs into G for encoding $\zeta(i, j)$. Regardless of the value of W, increment in the bitstream size is suppressed as soon as $SO_{\max}$ increases, and the effect becomes more obvious for larger $SO_{\max}$. The increment caused by using $SO_{\max} = 8$ is, on average, half of the increment caused by using $SO_{\max} = 1$. This verifies the ability of the proposed scanning order in suppressing bitstream size increment. In some cases, the output bitstream size is actually smaller (i.e., negative values) than its original coun-

terpart. However, we emphasize that this is due solely to the statistics of the image. Inevitably, the output bitstream size further increases as W increases due to the inefficiency of DPCM in predicting non-correlated neighboring DC components. To estimate the increment caused solely by scrambling the DC components, the increment caused by $W = 1$ is subtracted from the increment caused by the W value in question. In case of Lenna using $SO_{\max} = 8$, the increment caused by scrambling DC components with $W = 16$ is approximately $5.36\% - 1.47\% = 3.89\%$ over the increment caused by scrambling DC components with $W = 1$. It is also observed that an image of higher spatial activity (e.g., Baboon) has a lower percentage of increment because the ratio of $|L_2|$ to $|L|$ is relatively small when compared to an image of lower spatial activity.

Finally, we verify the consistency of SURIH in terms of embedding capacity and bitstream size increment using a large dataset. To this end, we consider the CalTech101 image dataset (i.e., 9146 images in total, with various dimensions) (Fei-Fei et al., 2007). We first convert each image into a grayscale image. We then collect the representative results obtained by applying our proposed SURIH with $n = 2$, $SO_{\max} = 1$, and $W = 1$ on all CalTech101 images. We choose these three parameters to ensure consistent and fair comparison between the CalTech101 results and the results obtained from the six standard test images. **Figure 4.6** summarizes the results for embedding capacity in terms of bpac (bits per nonzero coefficients). Results suggest that the embedding capacity centers around 0.15 bpnc, which is consistent with the results for the 6 standard test images considered (i.e., the results of $\Omega(G,2)/|L|$ in **Table 4.3**). **Figure 4.7** shows the distribution of percentage of bitstream size increment. It is observed that bitstream size is always increased, and the increment is tabulated around 5.7%, which is also consistent with the results recorded in the third column (i.e., $W = 1$) of **Table 4.5**. Therefore, we conclude that, on average, SURIH achieves ~ 0.15 bpnc with $\sim 5.7\%$ in bitstream size increment. Nevertheless, as verified earlier, bitstream size increment can be suppressed by consider-

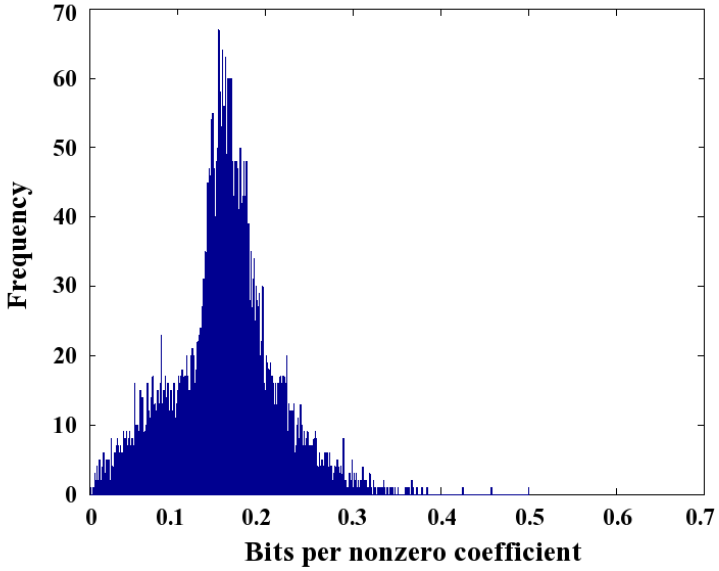


Figure 4.6: Carrier capacity [bits per nonzero coefficient] for CalTech101 image dataset (Max = 0.5004, Min = 0.0017, Mean = 0.1539, Median = 0.1551, STD = 0.0548)

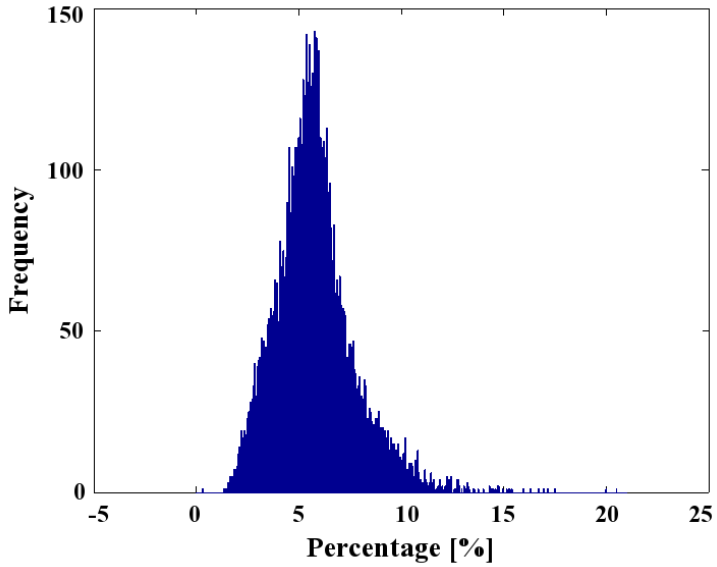


Figure 4.7: Percentage of bitsteam size increment [%] for CalTech101 image dataset (Max = 20.5164, Min = 0.3056, Mean = 5.7749, Median = 5.6120, STD = 1.9225)

ing larger $SO_{\max}$ and embedding capacity can be further increased by considering $n > 2$.

4.5.2 Performance w.r.t. Compression Quality Factor

Each raw, uncompressed, original test image is compressed at different quality factors, i.e., QF $\in \{30, 35, \cdots, 95\}$, to further capture the performance of SURIH. **Figure 4.8** shows the gross embedding capacity of each image w.r.t. various QF's. Here, we set

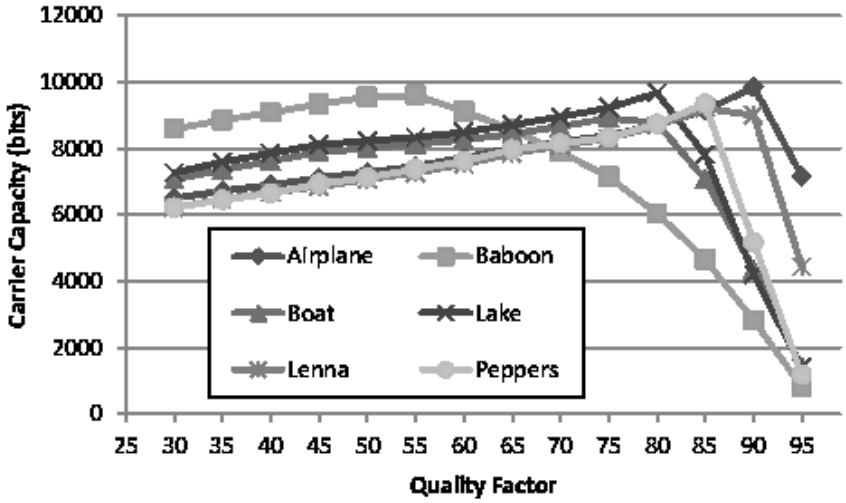


Figure 4.8: Gross embedding capacity w.r.t. various QF's

$n = 2$, $SO_{\max} = 1$ and $W = 1$. As expected, the result differs from image to image, but a general trend is observed. That is, as QF increases, the embedding capacity increases to its peak and drops to a small value when QF reaches 95. Nevertheless, most images reach their peaks at QF ~ 80, which is a commonly utilized QF in the JPEG compression standard. The result suggests that the embedding capacity peaks at a lower QF (e.g., 55) for an image of higher spatial activity (e.g., Baboon) because there are many ZRV pairs at higher QF. In particular, the average number of nonzero coefficients per block $\bar{\zeta}$ increases as QF increases (due to finer quantization), which results in a smaller number of practically achievable decompositions in VQD to embed external data. For example, $68322/64/64 \sim 16.7$ and $149590/64/64 \sim 36.5$ ZRV pairs are put into each 8×8 coefficient block of the Baboon image for QF$=$ 55 and 90, respectively. On the contrary, an image of lower spatial activity peaks at a larger QF (e.g., 90 for Airplane) because the number of ZRV pairs is low (i.e., small $|L|$) to realize high embedding capacity when QF is relatively low.

Next, we consider the percentage of the bitstream size increment w.r.t. various QF's and the results are shown in **Figure 4.9**. DC components are deliberately unprocessed

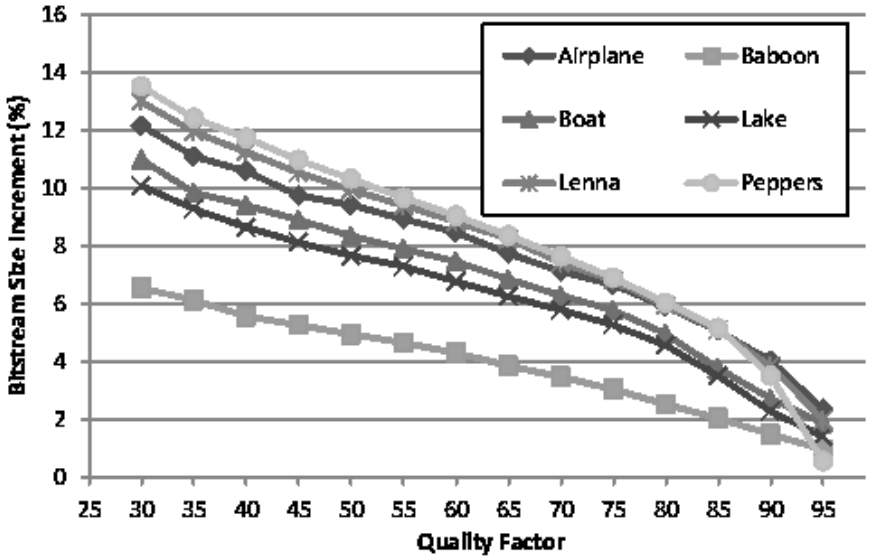


Figure 4.9: Percentage of bitstream size increment w.r.t. various QF

(i.e., $W = 1$) so that the effect of QF on bitstream size increment can be easily studied. Nevertheless, the percentage of increment in bitstream size attributed to scrambling DC component could be estimated as described earlier. It is observed that the percentage of the bitstream size increment decreases as QF increases. This is due to the fact that the same number of foreign ZRV pairs, i.e., $|L_2| = M \times N$ of them, are added to the image regardless of QF's. Since the ratio of $|L_2|$ to $|L_1|$ decreases as QF increases, the percentage of the bitstream size increment also decreases. For the same reason, an image of higher spatial activity (e.g., Baboon) tends to have a smaller percentage of bitstream size increment when compared to a relatively smooth image (e.g., Airplane) for a specific QF.

4.5.3 Comparison with Existing Methods

For fair comparison purposes, we only compare to existing methods that can completely reconstruct the original JPEG image.[4] In particular, we consider the bitstream

[4]We did not consider the unified method (Lian et al., 2007) because: a) it operates in the compressed video domain where the quality degradation effect is achieved partly by randomizing the prediction mode in INTRA-frame, which is not available in a still image; b) its implementation of randomizing sign of coefficients cannot withstandthe attacks mentioned in (W. Li & Yuan, 2007) thus the sketch of the original image can be revealed immediately; c) its data embedding mechanism is not reversible. We did not consider

Table 4.6: Percentage of bitstream size increment [%] and embedding capacity [bits] for the existing methods considered

	Airplane	Baboon	Boat	Lake	Lenna	Peppers
FIBS (W. Li & Yuan, 2007)	23.55	9.77	14.83	15.09	19.34	16.22
T1 (Takayama et al., 2006)	3.76	1.33	2.34	3.08	3.94	4.18
$\gamma^{\dagger}$	Scalable Scrambling by Takayama et al. (Takayama et al., 2008)					
2	0.06	0.13	0.11	0.10	0.08	0.16
4	16.27	16.76	18.93	17.19	19.29	20.41
8	46.52	47.86	49.05	47.88	52.12	54.37
16	82.19	86.41	85.77	84.83	89.70	92.40
32	116.49	125.91	121.02	120.63	125.15	128.41
64	178.06	194.81	182.99	184.75	187.41	191.69
128	213.68	233.32	218.57	221.25	223.00	227.98
256	259.69	283.42	265.13	268.57	268.02	273.57
	Histogram Pair by Xuan et al. (Xuan et al., 2009)					
$\Omega^{\ddagger}$ (bits)	5.84 (14098)	2.28 (10002)	5.11 (14410)	5.05 (15257)	5.03 (12742)	4.63 (12707)
	Lin et al. (C.-Y. Lin et al., 2008)					
$\Omega^{\ddagger}$ (bits)	14.20 (15160)	2.88 (11228)	9.29 (15702)	9.91 (15982)	41.34 (14975)	12.91 (15036)
$\oplus$	9.57	3.59	7.32	8.13	8.90	8.72
SURIH*	10.17	3.75	7.58	7.87	10.43	10.50

$^{\dagger}\alpha$ in (Takayama et al., 2008) is referred to as γ here

‡Actual embedding capacity achieved

*The largest bitstream size increment observed in SURIH

Table 4.7: SSIM (10^{-3}) and PSNR (dB) for Takayama et al.'s scalable scrambling method

γ	Airplane	Baboon	Boat	Lake	Lenna	Peppers
2	224 (14.7)	041 (11.0)	131 (14.2)	234 (13.0)	130 (16.1)	232 (15.6)
4	197 (14.5)	036 (10.7)	108 (13.9)	200 (12.8)	113 (15.9)	195 (15.4)
8	131 (13.9)	023 (9.7)	065 (12.7)	125 (12.0)	077 (14.8)	118 (14.5)
16	068 (12.1)	012 (8.0)	029 (10.3)	059 (10.0)	042 (12.4)	053 (12.1)
32	032 (9.5)	006 (6.8)	013 (8.1)	024 (7.8)	021 (9.6)	022 (9.3)
64	016 (7.5)	004 (6.4)	007 (6.8)	011 (6.5)	012 (7.7)	010 (7.3)
128	010 (6.3)	004 (6.3)	005 (6.3)	006 (5.9)	007 (6.6)	006 (6.3)
256	007 (5.7)	005 (6.3)	005 (6.2)	004 (5.7)	004 (6.2)	005 (6.0)
Range	217 (8.9)	037 (4.7)	126 (8.0)	125 (7.3)	230 (10.0)	228 (9.7)

size increment for images scrambled by Li's method (W. Li & Yuan, 2007) (referred to as FIBS) and both Takayama's methods (Takayama et al., 2006, 2008) (hereinafter referred to as T1 and T2, respectively). At the same time, we consider bitstream size increment for images embedded with external data by using Xuan's method (Xuan et al., 2009)

the reversible unified hiding method (X. Zhang, 2012) either since it operates in the spatial domain.

and Lin's method (C.-Y. Lin et al., 2008). Here, we have tuned the parameters in each reversible data embedding method to achieve approximately the same gross embedding capacity as obtained by SURIH with $n = 5$. The results are recorded in **Table 4.6**.

As expected, bitstream size always increases when an image is scrambled. FIBS and T2 (using $\gamma > 4$ for sufficient distortion) both produce a larger bitstream size than SURIH although they cannot accommodate any external data. In particular, T2 (Takayama et al., 2008) suffers from severe increment in bitstream size due to the significant change of magnitude in each nonzero coefficient (i.e., small $\leftrightarrow$ big), especially when γ is large. Although T1 produces smaller bitstream size increment when compared to SURIH, T1 is neither able to accommodate any external data nor achieve scalability in quality degradation.

The bitstream size is also increased due to data embedding. In particular, when embedding approximately the same amount of information (as recorded in parenthesis), Lin's method (C.-Y. Lin et al., 2008) achieves comparable bitstream size increment with that of SURIH. On the other hand, Xuan's method (Xuan et al., 2009) is consistently of smaller bitstream size increment when compared to both SURIH and Lin's method (C.-Y. Lin et al., 2008). However, it should be noted that both Xuan's and Lin's methods (Xuan et al., 2009; C.-Y. Lin et al., 2008) do not offer the scrambling feature. In fact, the embedding capacity in (C.-Y. Lin et al., 2008) and (Xuan et al., 2009) can be further increased, but at the expense of a larger output bitstream size. On the contrary, the gross embedding capacity in SURIH can be further increased (i.e., by increasing n) without causing increment in bitstream size. Also, SURIH can further suppress the bitstream size increment by considering more scanning orders (i.e., increasing $SO_{\max}$).

To further compare SURIH with the existing methods, we merged Takayama's T1 method (Takayama et al., 2006) and Xuan's method (Xuan et al., 2009) to embed external data into an image and scramble the output. We simply denote this method by $\oplus$, and

the results are recorded at the second to the last row in **Table 4.6**. Here, we compare the performance of $\oplus$ to the largest bitstream size increment observed in SURIH. Although SURIH and the $\oplus$ algorithm have comparable bitstream size increment, $\oplus$ has no control on the perceptual fidelity (i.e., scalable functionality in quality degradation). In addition, SURIH simultaneously handles both scrambling and data embedding operations, i.e., following the STE approach. However, $\oplus$ has to keep track of the order of operations to correctly extract the embedded data and to reconstruct the original image.

Last but not least, we compare the scalable quality degradation effect achieved by SURIH and T2 (Takayama et al., 2008). Again, SSIM and PSNR are considered as the evaluation metrics for image quality, and the results of T2 are recorded in **Table 4.7** for various values of γ. Note that we should not directly compare the results in **Table 4.4** and **Table 4.7** by matching the parameter values (say $W = \gamma$) because distortion is achieved by different means for each method. Instead, we should look at the range of SSIM values achieved. The range for SURIH (i.e., $\max - \min \sim 0.572$) is wider than that of T2 (i.e., ~ 0.230), implying that SURIH has greater flexibility in controlling the distortion level. Similar arguments are applicable to the observed PSNR values. Furthermore, the QT entries in SURIH can be tuned to achieve different types of distortion while this option is not available in T2 because it has no data embedding functionality to store the original QT entries for perfect reconstruction. More importantly, T2 fails to withstand the sketch attack in (Minemura et al., 2012) even when the image is distorted to the more severe level with respect to T2.

4.6 Summary

In this chapter, SURIH is proposed to seamlessly unify two disciplines of information hiding, namely, scrambling and reversible data embedding, in JPEG compressed images using the STE approach. SURIH is able to achieve scalable quality degrada-

tion in scrambling and scalable embedding capacity in data embedding at the same time. Specifically, an adaptive scanning order-based pre-processing technique is first proposed to choose the optimum scanning order to suppress the bitstream size increment. Then, a novel VQD data representation scheme is proposed to embed external data to achieve scalable embedding capacity without further causing bitstream size increment. Finally, we proposed to permute AC coefficients without restricting them to a particular block or subband and shuffle the DC coefficients within predefined blocks to progressively intensify distortion. Performance of SURIH is verified using various standard test images and all 9146 images from the CalTech 101 dataset. We also compared the proposed SURIH with existing scrambling methods, reversible data embedding methods, and their combinations. The extensive experiments demonstrated that the proposed SURIH offers more functionalities than the existing and the combined methods for approximately the same bitstream size increment and the same embedding capacity.

CHAPTER 5

UIH BASED ON NATURAL IMAGE PROPERTIES (NIP) IN SPATIAL AND COMPRESSED DOMAINS

5.1 Introduction

In this chapter, novel unified information methods following the S2E approach are proposed in spatial domain and JPEG compressed domain, aiming to achieve scrambling and external data insertion simultaneously. The chapter is divided into two major sections, namely: (a) UIH method following the S2E approach in spatial domain (i.e., hereinafter referred to as NIPS), and (b) UIH methods following the S2E approach in JPEG compressed domain (i.e., NIPC).

In **Chapter 3**, HAM method which follows the E2S approach, is proposed to scramble-embed the image in spatial domain. Nevertheless, output of HAM results in blockiness due to its block-based operation. Therefore, the first part of this chapter proposes NIPS method that operates on rows and columns. These row-based and column-based techniques are able to smoothen the uneven distortion of the output when achieving scalability. The properties in spatial domain (i.e., pixel) are exploited for image recovery and external data insertion purposes. The proposed method is detailed in **Section 5.2**.

In addition, previous chapter realized scrambling-embedding in JPEG compressed domain using the STE approach. In particular, the STE approach is deployed on AC coefficients while only scrambling is performed on DC coefficient. For SURIH, the order of decoding process are restricted and the bitstream size increment for standard test images are approximately 8.4%. Therefore, the second part of this chapter is to propose an UIH method which persues the S2E approach to further decrease the bitstream increment and fully utilize all the coefficients (AC and DC coefficients) for scrambling-embedding

purposes while achieving scalability. The properties in DC coefficients, energy of AC coefficients block, and run of zero AC coefficients are exploited. Two techniques are proposed to degrade the perceptual quality while manipulating its coefficients for data embedding. Experiments are conducted to measure the performance of the proposed unified method by using the UCID (Uncompressed Color Image Database) database and standard test images. The proposed method are also compared with the conventional methods in terms of effective capacity, image quality degradation, bitstream size increment and robustness against multiple sketch attacks. All the aforementioned details are discussed in **Section 5.6**.

5.1.1 Objectives

This chapter proposes UIH methods in spatial (namely, NIPS) and JPEG compressed (namely, NIPC) domains by using scrambling techniques while embedding data into the image to achieve scalability and reversibility (i.e., S2E approach). Natural image properties in both spatial and JPEG compressed domains are explored and exploited to propose UIH methods. In spatial domain, the existing problems to solve include:

1. Eliminate blockiness and edginess occurred in HAM in **Chapter 3**, and
2. Enhance scope of scrambling by introducing permutation method in HAM.

On the other hand, in JPEG compressed domain, the existing problems to solve include:

1. Further suppress bitstream size increment as observed in SURIH (**Chapter 4**), and
2. Fully utilize all the components in JPEG for scrambling-embedding.

5.2 NIPS: The Proposed Scrambling-Embedding (SE) Method

5.2.1 Introduction

This section proposes a UIH method called NIPS to realize scrambling-embedding in spatial domain by following the S2E approach. The rows and columns of pixels are first divided into groups. Rows are rotated to the left group-by-group to distort the image. Every rotation to the left results in a unique state, which in turn utilized to represent external information. For columns, the groups are rotated to the bottom. Similarly, unique states are created through rotation and they are utilized to embed information. Rows and columns can be restored and external data can be extracted using the discovered image property. By using different parameter values, the quality of the output image can be gradually degraded.

The proposed encoding and decoding processes are detailed in **Section 5.2.2** and **Section 5.2.3**. **Section 5.3** discusses the side information and the integration with previous HAM. Experiments are conducted and the results are illustrated in **Section 5.4**. **Section 5.5** summarizes this section.

5.2.2 Scrambling and Data Embedding

Suppose a grayscale image G has $8M \times 8N$ pixels and let $G(\mathfrak{m},\mathfrak{n})$ denote the pixel value at position $(\mathfrak{m},\mathfrak{n})$ for $1 \leq \mathfrak{m} \leq 8M$ and $1 \leq \mathfrak{n} \leq 8N$. In general, the neighboring pixels in natural image are highly correlated, i.e.,

$$G(\mathfrak{m},\mathfrak{n}) \sim G(\mathfrak{m}+e_{\mathfrak{m}},\mathfrak{n}+e_{\mathfrak{n}}), \tag{5.1}$$

for small $e_{\mathfrak{m}}$ and $e_{\mathfrak{n}}$ (i.e., slight difference with the adjacent pixels). This correlation is utilized as the basis (property) for the descrambling and data extraction purposes in the proposed method.

The encoding process (i.e., scrambling and data embedding) is performed by rotating

one row of pixels at a time until all rows are considered. The same process can be applied in the column manner.

In the row encoding process, let $row_{\mathfrak{m}}$ denote the $\mathfrak{m}$-th row of pixels in the image (from the top) for $1 \leq \mathfrak{m} \leq 8M$. The first row of the image, i.e., row_1, is left unmodified and the encoding process is applied for $2 \leq \mathfrak{m} \leq 8M$. Pixels in each row are divided into groups where each group consists of P_H pixels for $1 \leq P_H \leq 8M$. Each group is referred to as A_η for $\eta \in [1, 8N/P_H]$. These groups are rotated to the left for distorting the image while maintaining their relative order within a row to create unique states. The number of achievable states in a row is denoted by ρ and it is obtained by using:

$$\rho = 8N/P_H \times 2. \tag{5.2}$$

Here, $8N/P_H$ is multiplied by 2 because the row can be first reversed then divided into groups. In particular, reverse refers to the process of turning the row over in the left-to-right manner, i.e., first-in-last-out order. Hence, the original and reversed rows generate ρ distinct states altogether. Each state is denoted by R_ι for $\iota \in [1, \rho]$ and two consecutive states are related as follows:

$$R_\iota = R_{\iota-1} \ll P_H \tag{5.3}$$

where R_1 is the original row for $\iota \in [2, \rho/2]$, and $R_{(\rho/2)+1}$ is the reserved row utilized to find R_ι in the range $\iota \in [\rho/2) + 2, \rho]$. Here, the expression $\ll P_H$ refers to the left shift operation by P_H pixels, i.e., removing the first group on the left and putting it on the far right. Each of the state is utilized to carry $\lfloor \log_2 \rho \rfloor$ bits of the external information.

Figure 5.1 illustrates the grouping and rotation processes during encoding. A row of $8N = 32$ pixels is divided into 4 groups (denoted by A_1, A_2, A_3, and A_4) thus P_H = 8. In this case, 8 states can be generated based on **Equation (5.3)**. **Table 5.1** shows all

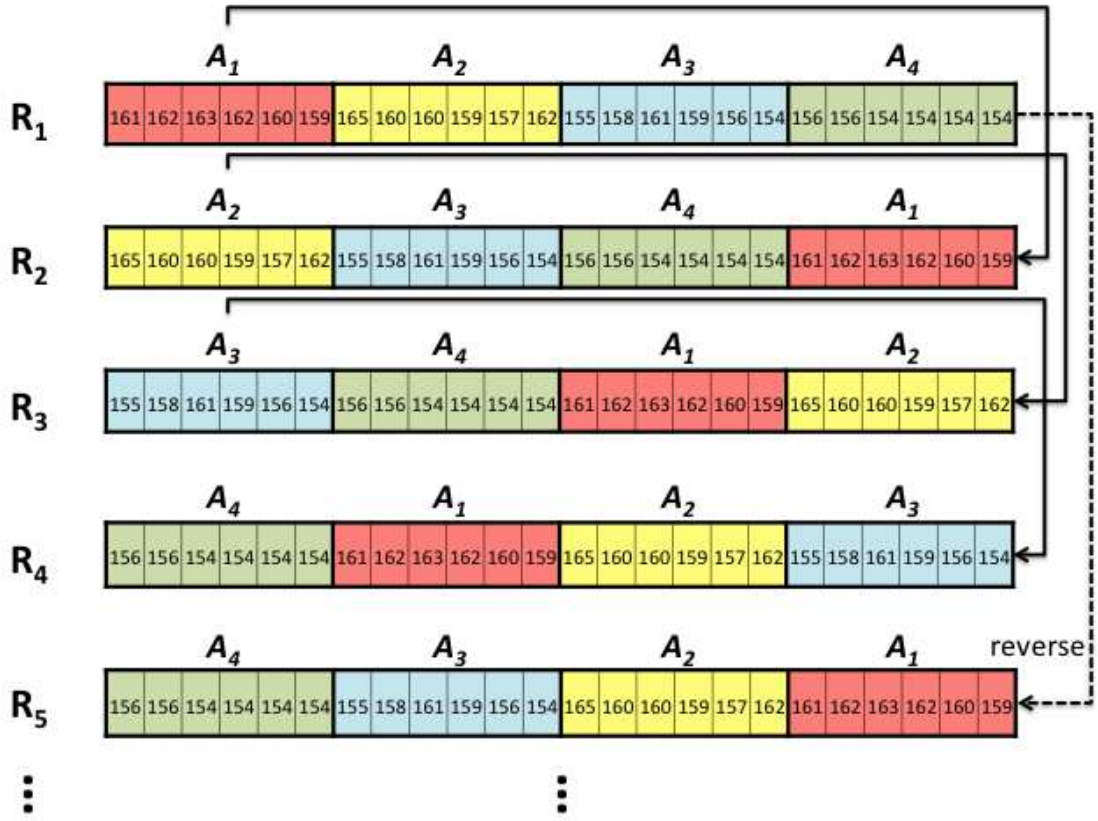
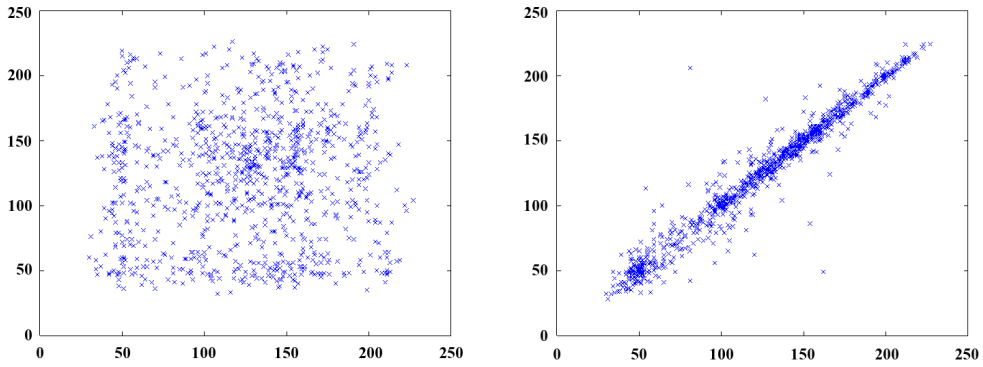


Figure 5.1: Grouping and rotating steps in encoding process when $P = 8$.

Table 5.1: States for example in **Figure 5.1**

R_i	States	Encoded Bits
R_1	$\{A_1, A_2, A_3, A_4\}$	000
R_2	$\{A_2, A_3, A_4, A_1\}$	001
R_3	$\{A_3, A_4, A_1, A_2\}$	010
R_4	$\{A_4, A_1, A_2, A_3\}$	011
R_5	$\{A_4, A_3, A_2, A_1\}$	100
R_6	$\{A_3, A_2, A_1, A_4\}$	101
R_7	$\{A_2, A_1, A_4, A_3\}$	110
R_8	$\{A_1, A_4, A_3, A_2\}$	111

possible outcomes of the rotation and reversing processes for the same example. Each of the state is utilized to store 3 bits of information. For instance, if the external information is '011', the row is rotated using **Equation (5.3)** for three times. In other words, the row is replaced by $row_m \leftarrow R_4 = \{A_4, A_1, A_2, A_3\}$. In addition, the bit sequence encoded by every state (such as those in **Table 5.1**) can be permuted to avoid unauthorized extraction of the embedded information.



(a) Vertical Correlation (b) Horizontal Correlation

Figure 5.2: Horizontal and vertical correlations after row encoding process

The encoding process detailed above is repeated for each row in the image to generate the scrambled-embedded output image denoted as G'. **Figure 5.2(a)** and **(b)** illustrates the scatter plot which is generated by randomly selecting 1000 pixels in G'. Here, the graph in **Figure 5.2(a)** and **(b)** are produced by plotting $G(\mathfrak{m},\mathfrak{n})$ versus $G(\mathfrak{m}-1,\mathfrak{n})$ and $G(\mathfrak{m},\mathfrak{n})$ versus $G(\mathfrak{m},\mathfrak{n}-1)$, respectively. The purpose of the plots is to verify the pixel correlations in G' for the vertical and horizontal directions. As expected, the correlations between pixels in the vertical direction (i.e., pixels residing at the same column but from two consecutive rows) in G' is destroyed due to the rotation process in rows. However, the horizontal correlations remain high as suggested by **Figure 5.2(b)**. Therefore, column encoding process is performed to further demolish the correlations among pixels in the image, i.e., to further degrade the image while embedding more information into it.

The column encoding process is similar to the row encoding process described above. Let $col_{\mathfrak{n}}$ be the $\mathfrak{n}$-th column in the image for $1 \leq \mathfrak{n} \leq 8N$. The first column col_1 is left unchanged and the column encoding process is applied for $2 \leq \mathfrak{n} \leq 8N$. Again, the pixels from a column are divided into groups where each group consists of P_V pixels in total. Each group of pixels is denoted by A_η for $\eta \in [1, 8M/P_V]$. Again, these groups are rotated to the bottom to distort the image while maintaining their relative order within a column. Note that it is possible that $P_V \neq P_H$. The rest of the encoding (scrambling and data em-

bedding) steps are exactly the same as described for rows and we omit the discussions here.

Note that the order of row and column operations does not matter in the encoding process. However, the decoding process must follow the opposite order as in the encoding process to perfectly reconstruct the original image and extract the embedded information perfectly.

5.2.3 Descrambling and Data Extraction

The correlation expressed by **Equation (5.1)** is utilized in the decoding process (i.e., descrambling and data extraction). If **Equation (5.1)** holds true, the original state $R_{\iota'_0}$ of a row will yield the smallest sum of absolute differences (in pixel value) with respect to the previous row. Therefore, each row row'_m is divided into groups $A'_1, A'_2, \cdots, A'_{8N/P_H}$, and all possible states $R'_{\iota'}$ are considered where $R'_{\iota'} = \{A'_1, A'_2, \cdots, A'_{8N/P_H}\}$ for $\iota' = 1$. The sum of absolute differences $\Delta\varepsilon^V_{\iota'}(\mathfrak{m})$ between the pixels in $row'_{\mathfrak{m}}$ and $row'_{\mathfrak{m}-1} = row_{\mathfrak{m}-1}$ is calculated as:

$$\Delta\varepsilon^V_{\iota'}(\mathfrak{m}) = \sum_{\mathfrak{n}=1}^{8N} \mid G'_{\iota'}(\mathfrak{m},\mathfrak{n}) - G(\mathfrak{m}-1,\mathfrak{n}) \mid, \tag{5.4}$$

for $\mathfrak{m} \geq 2, G'_{\iota'}(\mathfrak{m},\cdot) \leftarrow R'_{\iota'}$ and $\iota' \in [1,\rho]$ to determine the original state $R'_{\iota'_0}$, which yields the smallest $\Delta\varepsilon^V_{\iota'}(\mathfrak{m})$. The original state can be found deterministically because the reserving process maintains the relative order among the groups, except for changing the sequences to first-in-last-out (FILO) or first-in-first-out (FIFO). Therefore,

$$\{R_1, R_2, \cdots R_\rho\} = \{R'_1, R'_2, \cdots R'_\rho\}, \tag{5.5}$$

which means all possibles states achievable during encoding can be achieved too during decoding.

Once the original state ι'_0 is identified from the rotated row $R'_{\iota'}$, the embedded information can be extracted by first identifying the current state using:

$$\iota = \begin{cases} 1 & \text{if } \iota'_0 = 1 \\ (\rho/2) - \iota'_0 + 2 & \text{if } 2 \leq \iota'_0 \leq (\rho/2) \\ (\rho/2) + 1 & \text{if } \iota'_0 = (\rho/2) + 1 \\ 2\rho - \iota'_0 + (2 - (\rho/2)) & \text{otherwise.} \end{cases} \tag{5.6}$$

By using the example given in the encoding process, the original state is identified to be $R'_{\iota'_0=2}$. Thus, **Equation (5.6)** generates the result of $\iota = 4$ when $\iota'_0 = 2$. In other words, the embedded data is the bit sequence represented by R_4 in **Table 5.1**, which is '011'. At the same time, the row is modified to its original state to enable the decoding process for the next row row'_{m+1}.

For the column decoding process, similar steps are applied and we omit the presentation here.

5.3 NIPS: Discussion

In this section, side information utilized in the decoding process are described in **Section 5.3.1** . In addition, **Section 5.3.2** discusses the integration with HAM (Ong, Wong, & Tanaka, 2012) to enhance the robustness and increase the capacity of NIPS.

5.3.1 Side Information

Preliminary analysis is done on six test images to verify the validity of **Equation (5.1)**. It is found that the correlation holds true most of the time. However, when the proposed method divides the rows (columns) into groups of P_H (P_V) pixels, the original state for some of the rows (columns) are not identifiable using the proposed decoding process. **Table 5.2** shows the average percentage of the unidentifiable rows and columns for various sizes of P_H and P_V. It is observed that when P_H and P_V decrease, the number of unidenti-

Table 5.2: Percentage [%] of the unidentifiable rows and columns for variable P

P	Unidentifiable Row	Unidentifiable Column
16	0.1	0.1
8	0.1	0.1
4	0.2	0.3
2	0.6	1.4

fiable rows and columns increase. In other word, the states are less distinguishable when the group size decreases (i.e., the number of groups increases).

Nevertheless, the percentage of unidentifiable rows and columns are relatively small because they amount for only $\sim$3 (0.6%) rows and $\sim$7 (1.4%) columns in the image for $P_H = B_V = 2$. However, pre-processing is needed to mark the unidentifiable rows and columns before the encoding process takes place. In particular, the location of the unidentifiable rows and columns are stored in a raw format (i.e., 9 bits per row or column for an 512$\times$512 image) as side information to ensure perfect image reconstruction and correct data extraction.

5.3.2 Robustness and Capacity Improvement using HAM

Permutation and substitution are the two fundamental building blocks for any encryption method (Shannon, 1949). In the proposed method, only the permutation operation is utilized to scramble and embed data. The decoding process is implemented by relying on the correlations among pixels in the image. Therefore, to further improve its robustness against unauthorized viewing or decoding of the embedded message, (Ong et al., 2012) is applied on top of the proposed method to substitute the pixel values. (Ong et al., 2012) propose to scramble the image by using histogram modification-based data embedding method. In addition, (Ong et al., 2012) can also be utilized to increase the payload of the proposed method.

(Ong et al., 2012) operates in a block basis to maximize the payload because higher correlation in a block provides more available spaces in histogram for data embedding.

Nevertheless, the output image from our proposed method is scrambled. Hence, instead of forming blocks of pixels as proposed in (Ong et al., 2012), $b_s \times b_s$ number of pixels are randomly (but not repeatedly) selected and grouped to form blocks of pixels using a key. Selected pixel values are substituted with another unutilized values in the histogram to embed more information and further distort the final output image quality.

In the decoding process, the operations in (Ong et al., 2012) must be first reversed (i.e., decoding) to enable correct image reconstruction and data extraction. This flow of process prevents the unauthorized party to recover the original image and the embedded data without the valid keys. Using the management scenario as mentioned in **Chapter 2**, the unauthorized viewer needs the valid key to restore the original pixel values because they are substituted by other values, i.e., modified by (Ong et al., 2012). On the other hand, the secretary can only access to the administrative metadata from column-based decoding process without being able to view the original image. Finally, a manager with the highest authority level has access to both the original image and the embedded confidential data after the row-based decoding operation.

5.4 NIPS: Experimental Results

Six standard grayscale test images (i.e., Airplane, Baboon, Boat, Lake, Lenna and Peppers) (*The USC-SIPI Image Database [online]*, n.d.) each with dimension 512×512 pixels are considered in the experiments. It is verified that the proposed method can degrade the input image with controllable distortion, while the original image and the embedded information can be perfectly reconstructed directly from the scrambled-embedded image using the correct keys.

Table 5.3 and **Table 5.4** records the SSIM (structural similarity index measurement (Z. Wang et al., 2004)) and PSNR (pixel signal-to-noise ratio) values for different ranges of $\omega_{\max}$ when $P = P_H = P_V = 8$. Parameter $\omega_{\max}$ is utilized to restrict the number

Table 5.3: Image Quality (SSIM) for various $\omega_{\max}$ when $P = 8$

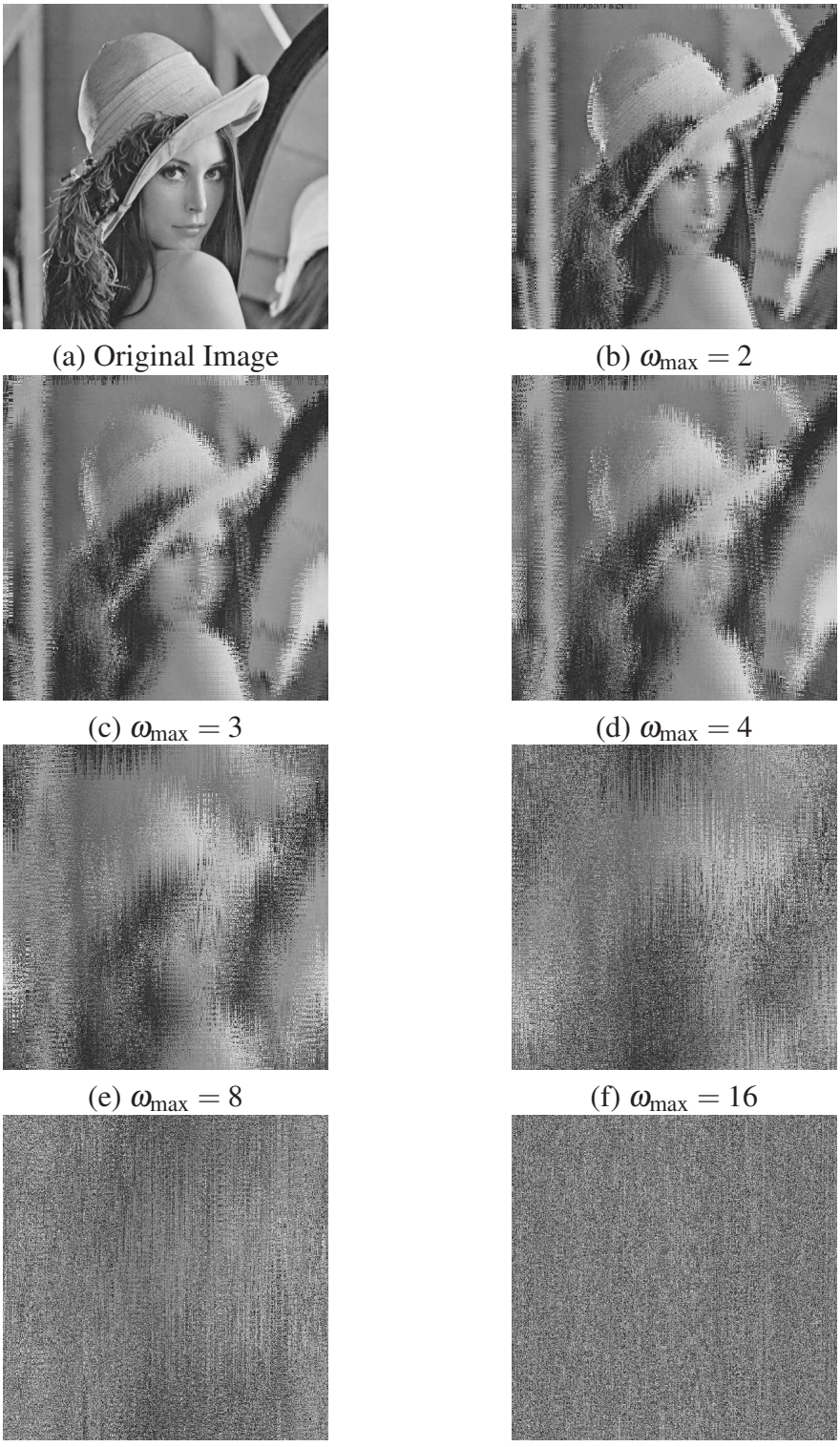
$\omega_{\max}$	Airplane	Baboon	Boat	Lake	Lenna	Peppers	Average
2	0.526	0.279	0.389	0.385	0.453	0.427	0.410
4	0.342	0.117	0.191	0.188	0.216	0.194	0.208
8	0.224	0.055	0.105	0.088	0.081	0.070	0.104
16	0.116	0.035	0.069	0.036	0.038	0.028	0.054
32	0.046	0.026	0.043	0.015	0.026	0.020	0.029
64	0.029	0.026	0.029	0.011	0.024	0.019	0.023
128	0.029	0.026	0.031	0.011	0.024	0.019	0.024

Table 5.4: Image Quality (PSNR) for various ω when $P = 8$

$\omega_{\max}$	Airplane	Baboon	Boat	Lake	Lenna	Peppers	Average
2	17.54	16.81	17.96	15.97	18.17	17.50	17.32
4	14.75	15.04	14.96	12.99	14.56	13.79	14.35
8	13.54	13.71	13.52	11.29	12.62	11.68	12.73
16	12.66	12.78	12.57	9.94	11.86	10.60	11.73
32	12.04	12.58	11.78	8.78	11.48	10.44	11.18
64	11.84	12.65	11.78	8.79	11.54	10.55	11.19
128	11.82	12.62	11.74	8.83	11.56	10.52	11.18

of times a row or column can be rotated. For $P = 8$, $\rho = 128$ based on **Equation (5.2)**. Therefore, the output quality can be adjusted by using $\omega_{\max} \in [1, 128]$. Based on the results, the output image quality is higher for both measurements when $\omega_{\max}$ is small but degrades gradually as $\omega_{\max}$ increases. On average, the highest SSIM (PSNR) is $\sim$0.410 ($\sim$17.32 dB) for $\omega_{\max} = 2$. On the other hand, the average lowest SSIM (PSNR) is $\sim$0.024 (11.18 dB), i.e., when the highest number of rotations $\omega_{\max} = 128$ is considered. These results, i.e., SSIM and PSNR are higher for smaller $\omega_{\max}$, are expected because there are less pixels involved in the rotation. For instance, $\omega_{\max} = 2$ rotates at most $2 \times P$ pixels while $\omega_{\max} = 16$ rotates at most $16 \times P$ pixels to the left (i.e., for row operation) or to the bottom (i.e., for column operation) during encoding.

Figure 5.3 shows the scrambled-embedded images of Lenna using various $\omega_{\max}$ when $P = 8$. By visual inspection, the output images are observed to be severely scrambled when $\omega_{\max} = 128$. In addition, the output images are gradually degraded, resembling



(a) Original Image (b) $\omega_{\max} = 2$

(c) $\omega_{\max} = 3$ (d) $\omega_{\max} = 4$

(e) $\omega_{\max} = 8$ (f) $\omega_{\max} = 16$

(g) $\omega_{\max} = 32$ (h) $\omega_{\max} = 128$

Figure 5.3: Original and output Lenna images for various $\omega_{\max}$ for $P = P_H = P_V = 8$

to the oil painting effect with increasing brush stroke size but no blockiness nor visible outline that appear in (Ong et al., 2012; Ong, Minemura, & Wong, 2013) when $\omega_{\max}$ increases, which agree with the results recorded in **Table 5.3** and **Table 5.4** .

Table 5.5: Image Quality (SSIM) for various P when $\omega_{\max} = 4$

P	Airplane	Baboon	Boat	Lake	Lenna	Peppers	Average
1	0.675	0.272	0.473	0.512	0.633	0.623	0.531
2	0.544	0.190	0.357	0.369	0.494	0.479	0.406
4	0.440	0.147	0.263	0.267	0.357	0.336	0.302
8	0.342	0.117	0.191	0.188	0.216	0.194	0.208
16	0.253	0.081	0.130	0.109	0.101	0.088	0.127
32	0.155	0.060	0.094	0.058	0.052	0.039	0.076
64	0.076	0.052	0.071	0.030	0.041	0.028	0.050
128	0.047	0.052	0.046	0.022	0.038	0.029	0.039

Table 5.6: Image Quality (PSNR) for various P when $\omega_{\max} = 4$

P	Airplane	Baboon	Boat	Lake	Lenna	Peppers	Average
1	20.93	17.48	20.67	19.61	22.43	21.97	20.51
2	18.28	16.60	18.44	16.90	19.41	18.85	18.08
4	16.26	15.82	16.54	14.74	16.78	16.07	16.04
8	14.75	15.04	14.96	12.99	14.56	13.79	14.35
16	13.83	14.05	13.88	11.64	12.97	12.06	13.07
32	13.05	13.14	12.98	10.47	12.24	10.97	12.14
64	12.37	12.78	12.26	9.390	11.82	10.62	11.54
128	12.04	12.98	11.99	8.962	11.76	10.77	11.42

Table 5.5 and **Table 5.6** gathers the SSIM and PSNR values for different P when $\omega_{\max} = 4$. The representative output image of Boat for **Table 5.5** and **Table 5.6** are illustrated in **Figure 5.4**. It is observed that P, i.e., the block size, can also be utilized as the control parameter to gradually distort an image. Both quality measurements indicate that the image quality is higher for smaller P, and vice versa. On average, the SSIM ranges from 0.531 (i.e., $P = 1$) to 0.039 (i.e., $P = 128$). For PSNR, the results are ranging from 20.51 dB (i.e., $P = 1$) to 11.42 dB (i.e., $P = 128$).

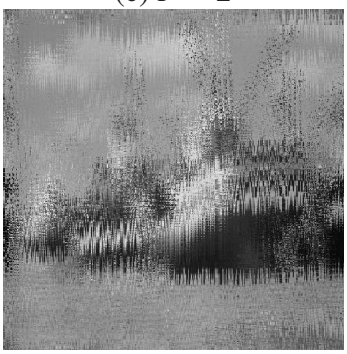
Next, the achievable payload is considered. **Table 5.7** shows the raw embedding capacity (i.e., no deduction of side information) for various $\omega_{\max}$. As expected, the embedding capacity is low when $\omega_{\max}$ is small. This is due to the limited number of achievable states for small $\omega_{\max}$. For example, $\omega_{\max} = 4$ enables only 4 states where each state encodes 2 bits, while $\omega_{\max} = 64$ allows 64 states where each state encodes 6 bits. Therefore,

(a) Original Image

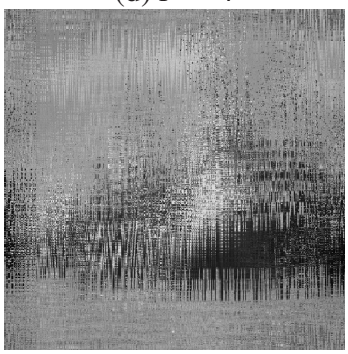
(b) $P = 1$

(c) $P = 2$

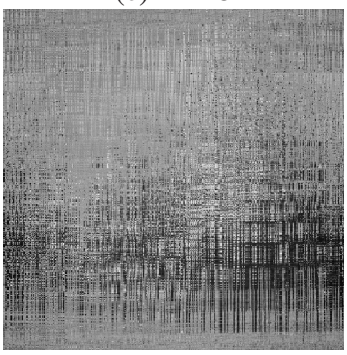
(d) $P = 4$



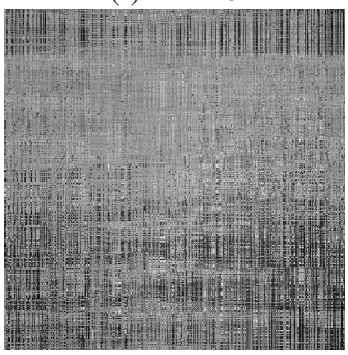
(e) $P = 8$



(f) $P = 16$



(g) $P = 32$



(h) $P = 64$

Figure 5.4: Original image and output Boat images for various P (i.e., P_H and P_V) when $\omega_{\max} = 4$

$\omega_{\max} = 128$ has the highest raw embedding capacity, which is 7154 bits. From the results shown in **Table 5.3**, **Table 5.4** and **Table 5.7**, we conclude that the quality decreases when more information is embedded.

Table 5.7: Average raw Embedding Capacity [Bits] for various $\omega_{\max}$ when $P = 8$

$\omega_{\max}$	Raw Embedding Capacity
2	1022
4	2044
8	3066
16	4088
32	5110
64	6132
128	7154

Table 5.8: Effective Embedding Capacity [Bits] for various P and citeConference:Ong2012

P	8	4	2	1	8 $\oplus$ (Ong et al., 2012)[1]
Airplane	7106	8125	9144	10144	13733
Baboon	7154	8142	8946	8662	38431
Boat	7154	8176	9198	9764	16670
Lake	7122	8142	9162	9897	64954
Lenna	7138	8074	8478	5812	71314
Peppers	7122	8142	9162	8225	20605
Average	7132.7	8133.5	9015.0	8750.7	37617.8

[1] $b_s = 16$ for (Ong et al., 2012)

Table 5.8 shows the effective embedding capacity (i.e., with the number of bits needed to store side information deducted) for various P. The results suggest that the effective embedding capacity increases when P decreases, and vice versa. This is because the number of states increases when P decreases. For instance, $P = 8$ has only 4 states as compared to $P = 2$ which produces 16 states for the specific case of $8N = 32$. However, the effective embedding capacity for $P = 1$ is lower than that of $P = 2$ because more side information is needed to store the location of the unidentifiable rows and columns for $P = 1$. Hence, the highest effective embedding capacity of $\sim$9015.0 bits is achieved by the proposed method when $P = 2$.

Next, (Ong et al., 2012) is applied on top of the proposed method to gain higher payload and increase robustness against unauthorized viewing of the original content.

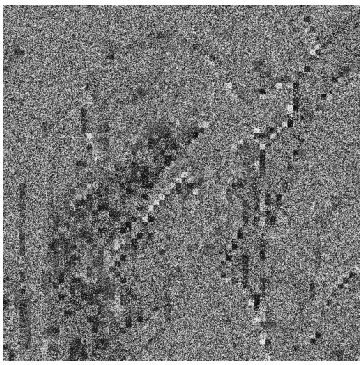
Table 5.8 shows the combined payload and suggests that the integration with (Ong et al., 2012) increases the capacity to ∼37617.8 bits. For completion of discussion, the proposed method is compared with (Ong et al., 2012) and (Ong et al., 2013) where their achievable effective payloads are recorded in **Table 5.9**. Note that (Ong et al., 2013) is designed for scrambling-embedding in the frequency domain and we reimplemented it in the pixel domain for comparison purposes. **Table 5.9** records the payload of the conventional methods considered. Results show that the proposed method outperformed (Ong et al., 2013) method in all block sizes. Also, block-rotation based scrambling-embedding method (Ong et al., 2013) is not suitable to be utilized in the pixel domain because it is challenging to identify the original rotation orientation due to the high correlations among the pixels around the rotated area. For example, for 8×8 block size, only ∼55.14% of the blocks are identifiable. A location map is created to record the identifiable and unidentifiable blocks which reduces the number of effective embedding capacity to ∼421.0 bits. Meanwhile, (Ong et al., 2012) achieves the highest embedding capacity among these three methods, which is ∼598,982 bits, on average. In addition, our method is also comparable to Zhang's method (X. Zhang, 2012), as the proposed method achieve 8478 bits embedding capacity and (X. Zhang, 2012) achieves $\sim$ 8650.8 bits in lossless mode.

Last but not least, both (Ong et al., 2012) and (Ong et al., 2013) are able to gradually degrade the image quality. However, both methods suffer from the blockiness and edginess artifacts as shown in **Figure 5.5(a)** and **(b)**, respectively. Furthermore, (X. Zhang, 2012) can only produce encrypted image (i.e., completely randomized) because the data embedding method is applied to the encrypted image to achieve scrambled-embedded output.

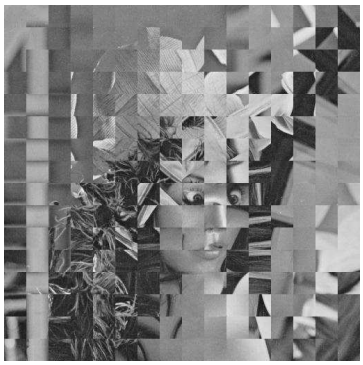
Table 5.9: Effective Embedding Capacity [Bits] Comparison

Method	Proposed	(Ong et al., 2013)			(Ong et al., 2012)
	Method[1]	32×32	16×16	8×8	4×4
Airplane	9144	142	466	976	779089
Baboon	8946	92	308	410	302074
Boat	9198	138	302	298	558594
Lake	9162	96	114	-226	539502
Lenna	8478	158	424	798	707319
Peppers	9162	120	258	270	707319
Average	9015.0	124.3	312.0	421.0	598982.8

[1] $P_H = 2$, $P_V = 2$ and $\omega_{\max} = 512$



(a) (Ong et al., 2012) (block size = 8×8)



(b) (Ong et al., 2013) (block size = 32×32)

Figure 5.5: Output Images for the conventional methods

5.5 NIPS: Summary

In this section, a reversible UIH method following the S2E approach was proposed to scramble an image and embed information into it. The proposed method is able to control the degradation in image quality in a smooth manner using the parameter $\omega_{\max}$ and P. Experimental results showed that the proposed method is able to embed up to ∼9015.0 bits on average. Another layer of robustness against unauthorized viewing can be added on top of the proposed method by adopting (Ong et al., 2012), which also further improve the effective payload to ∼37617.8 bits. The row-based and column-based methods utilized in this section are improved and implemented in the following section.

5.6 NIPC: The Proposed Scrambling-Embedding (SE) Methods

5.6.1 Introduction

In this section, a unified information hiding method following the S2E approach is proposed for the JPEG compressed image. Properties of the DC and AC coefficients are exploited for extraction of the embedded data and image reconstruction purposes. To the best of our knowledge, this is the first method that attempts to embed information through a scrambling process in the DCT domain. The proposed methods are designed to achieve scalability in embedding capacity and perceptual quality degradation while minimizing bitstream size increment.

First, the input image G with $8M \times 8N$ pixels is divided into 8×8 non-overlapping blocks $B(i, j)$, where $i \in \{1, 2, \cdots, M\}$ and $j \in \{1, 2, \cdots, N\}$. Each block is transformed into its frequency representation using DCT (Discrete Cosine Transformation) and the output is further quantized to produce the quantized coefficients. Each quantized coefficients within the block $B(i, j)$ is denoted by $B(i, j)_{u,v}$, where $u, v \in \{1, 2, \cdots, 8\}$. The upper left element, i.e., $B(i, j)_{1,1}$, is called the DC coefficient while the rest are called the AC coefficients. The AC coefficients are not coded directly because this approach will lead to minimal gain in terms of compression. Instead, the AC coefficients in a block are considered in zigzag order and they are run-length coded, where the count of zero coefficients before the occurrence of a non-zero coefficient (or the end of the block is reached) is considered (Pennebaker & Mitchell, 1992; *T.81 : Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines [online]*, 1992). The coefficients are encoded as a pair, i.e., (*zero run-length*, *actual non-zero AC coefficient*), by using the Huffman entropy coding. For the rest of the discussion, (*zero run-length*, *actual non-zero AC coefficient value*) is referred to as ZRV (zero-run value) pair.

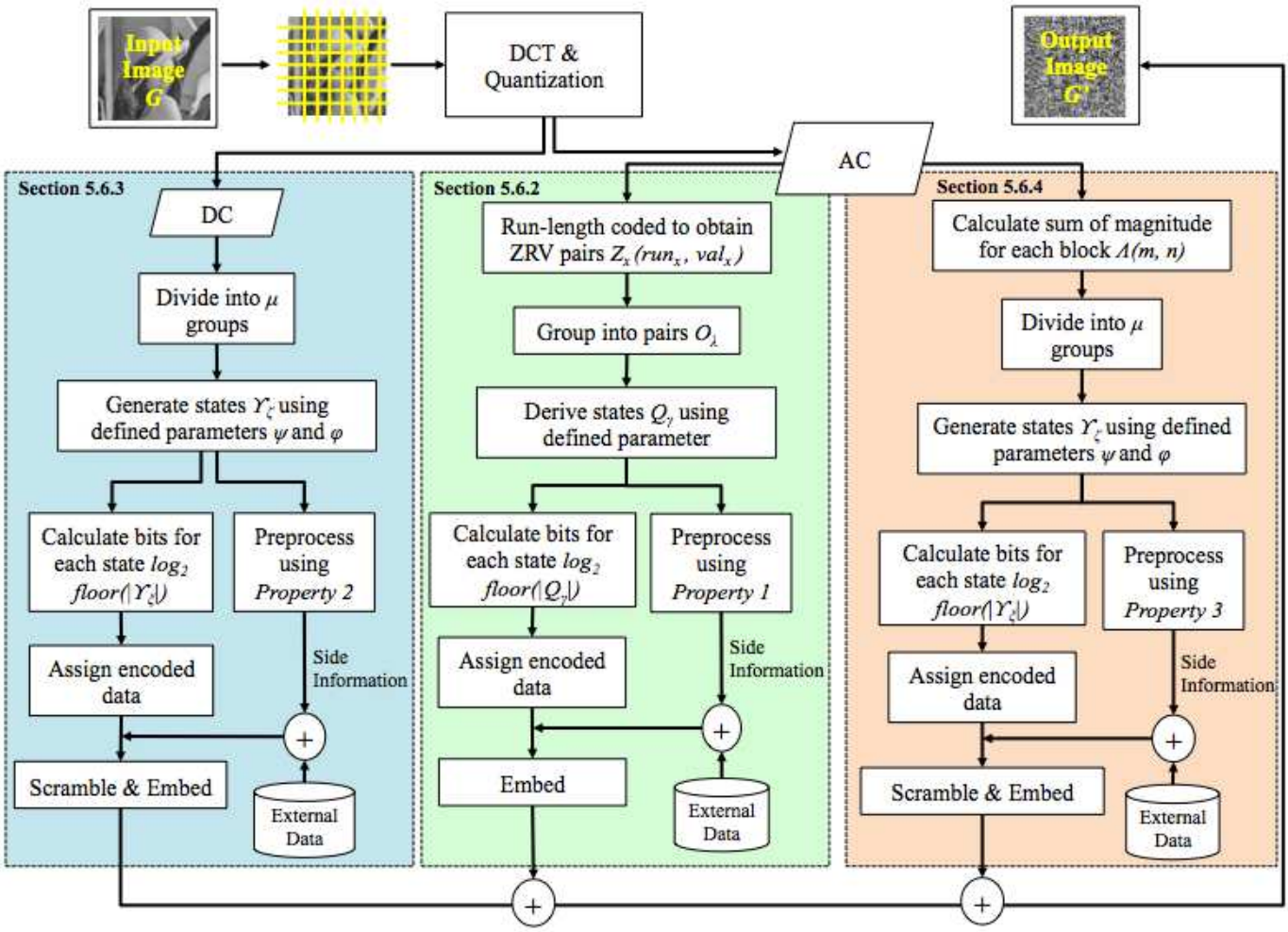


Figure 5.6: Encoding process for NIPC1, NIPC2 and NIPC3.

We exploit the properties in both DC and AC coefficients to realize scrambling-embedding. Here, two techniques are proposed and they are applied in three non-overlapping areas within a JPEG image. **Figure 5.6** shows the flowchart of the proposed method. First, we propose a scrambling-embedding technique by manipulating the ZRV pairs (i.e., NIPC1) in **Section 5.6.2**. Next, energy of AC coefficients (i.e., sum of magnitudes) and the DC coefficient for each block are collected into two $M \times N$ arrays. Another scrambling-embedding technique is proposed and applied on the DC coefficient array (i.e., NIPC2) and AC block energy array (i.e., NIPC3). The encoding and decoding processes for NIPC2 and NIPC3 are detailed in **Section 5.6.3** and **Section 5.6.4**, respectively. Possible applications and enhancements of the proposed methods are discussed in **Sec-**

tion 5.7. Experimental results are presented in **Section 5.8**, and **Section 5.9** summarizes this section.

5.6.2 SE in Zero-run Value Pairs and its Natural Property (NIPC1)

The recommended quantization table in the JPEG compression standard incorporates HVS (Human Visual Systems) to achieve a balanced rate-distortion trade-off (*T.81 : Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines [online]*, 1992). In particular, the visual information held by the high frequency sub-bands is less significant, perceptually, when compared to that held by the low frequency sub-bands. It is because human eyes are more sensitive to the overall (i.e., major) changes than the fine/subtle variations. Hence, the recommended quantization table utilizes small divisors for low frequency sub-bands to retain the perceptually significant information and large divisors for the high frequency sub-bands to remove the insignificant information. As a result, there are less zeros and larger (in terms of magnitude) AC coefficients in the low frequency sub-bands, while there are more zeros and smaller AC coefficients in the high frequency sub-bands. Thus, the AC coefficients in a block are then gathered in zigzag order (i.e., linearize a matrix into an vector) and run-length coded to produce the ZRV pairs.

Due to the aforementioned processes, ZRV pairs at the front end often have large magnitudes and short zero-runs while ZRV pairs at the back end have small magnitudes and long zero-runs. **Figure 5.7** records the statistics of zero-run and value (i.e., magnitude) for both the first and last ZRV pairs based on six standard test images (*The USC-SIPI Image Database [online]*, n.d.) (i.e., Airplane, Baboon, Boat, Lake, Lenna and Peppers) compressed at quality factor of 80. The zero-run part for the first ZRV pair is shorter (i.e., smaller) than that of the last ZRV pair for 73.9% of the time. In addition, the coefficient in the first ZRV pair has larger magnitude than that of the last ZRV pair for 82.1% of the

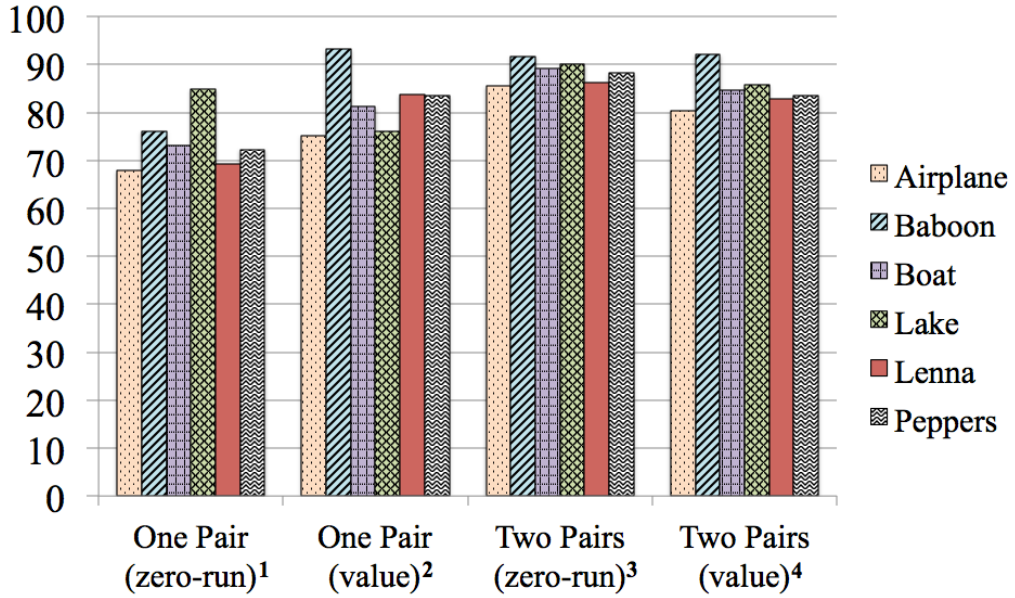


Figure 5.7: Statistics of ZRV properties [%] for six standard test images. [1]Zero-run of the first pair is smaller than zero-run of the last pair; [2]Magnitude of the first pair is larger than magnitude of the last pair; [3]Sum of zero-run for the first two pairs is smaller than the sum of zero-run for the last two pairs; [4]Sum of magnitude for the first two pairs is larger than the sum of magnitude for the last two pairs.

time. These trends are more obvious when considering the sum of two front ZRV pairs and the sum of two last ZRV pairs. For the same set of images, the percentages increase to 88.5% and 84.9% for the case of sum of zero runs and sum of magnitudes, respectively. Therefore, it is observed in JPEG that:

***Property 1**: The ZRV pairs in a quantized coefficient block have large magnitude and short zero-run for the front pairs while having small magnitude and long zero-run for the end pairs.*

This property is exploited in this section to realize scrambling-embedding. Suppose there are X number of ZRV pairs in a block, where each ZRV pair Z_χ contains two elements, namely zero-run run_χ and value val_χ for $\chi \in \{1,2,\cdots,X\}$. Due to the utilization of binary representation and limited number of ZRV pairs (i.e., $X \leq 63$), only 2^π ZRV pairs are considered in the encoding process for $\pi \in \{2,3,4,5\}$. Note that π is dependent on X

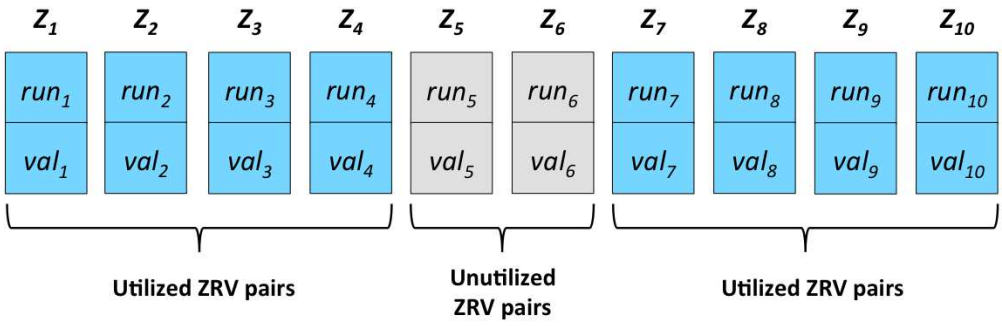
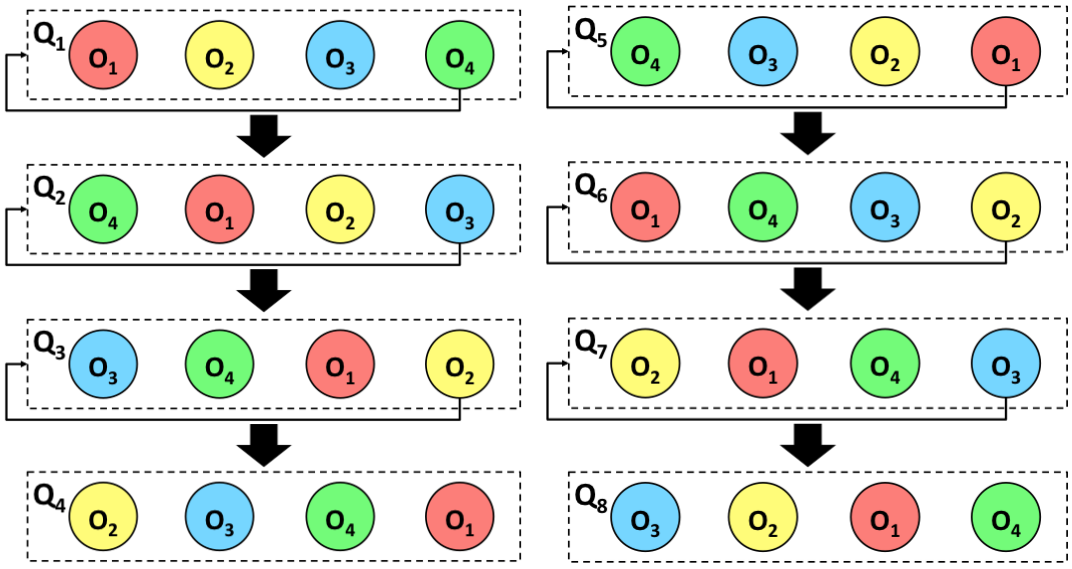


Figure 5.8: Utilized and unutilized ZRV pairs for $X = 10$



(a) Shifting from Q_1 to Q_4 (b) Shifting from Q_5 to Q_8

Figure 5.9: Illustration of ZRV groups shifting process for $d = 4$. Q_1 in (a) is flipped to produce Q_5 in (b).

in which case $\pi \leq \lfloor \log_2 X \rfloor$.

In the proposed method, only $d = 2^{\pi-1}$ ZRV pairs at the front and $2^{\pi-1}$ ZRV pairs indexed from the back are utilized, while the middle pairs are left unmodified. For instance, if $X = 10$, then $\pi = \lfloor \log_2 10 \rfloor = 3$. Here, the first four ZRV pairs and last four ZRV pairs are utilized, and the remaining two middle pairs are left unmodified as shown in **Figure 5.8**. To ease the presentation, we omit the middle pairs in the illustrations while showing only the chosen 2^{π} ZRV pairs.

Every two ZRV pairs are collected into a group and each group is denoted by O_λ for $\lambda \in \{1, 2, \cdots, d\}$. Using the same example given in **Figure 5.8**, the grouping process

Table 5.10: List of states for the example given in **Figure 5.9**

Q_γ	States	Encoded Bits
Q_1	$\{O_1, O_2, O_3, O_4\}$	000
Q_2	$\{O_2, O_3, O_4, O_1\}$	001
Q_3	$\{O_3, O_4, O_1, O_2\}$	010
Q_4	$\{O_4, O_1, O_2, O_3\}$	011
Q_5	$\{O_4, O_3, O_2, O_1\}$	100
Q_6	$\{O_3, O_2, O_1, O_4\}$	101
Q_7	$\{O_2, O_1, O_4, O_3\}$	110
Q_8	$\{O_1, O_4, O_3, O_2\}$	111

is illustrated in **Figure 5.9**, where $Z_1(run_1, val_1)$ and $Z_2(run_2, val_2)$ are grouped as O_1, $Z_3(run_3, val_3)$ and $Z_4(run_4, val_4)$ are grouped as O_2, and so forth. Groups in a block are shifted and flipped to create a total of $2d$ unique states for data embedding purposes. Here, each state is denoted by Q_γ for $\gamma \in \{1, 2, \cdots, 2d\}$. First, groups in a block are shifted using the following:

$$Q_\gamma = Q_{\gamma-1} \ll 1 \tag{5.7}$$

for $\gamma \in \{2, 3, \cdots, d\}$ and $\ll$ denotes the process of shifting one group to the left. Note that $Q_1 = \{O_1, O_2, \cdots, O_d\}$ is the initial state, i.e., the original position as in Q_1 of **Figure 5.9**. Example of Proposed Method I for $d = 4$ is shown in **Figure 5.9(a)**.

Next, the groups in a block are flipped to produce the states $Q_{d+1} = \{O_d, O_{d-1}, \cdots, O_1\}$, i.e., the indices are in decreasing order. From this particular state, the unique states of O_γ for $\gamma \in \{d+2, d+3, \cdots, 2d\}$ are produced by invoking **Equation (5.7)**. Using the same example, the flipping and shifting processes are illustrated in **Figure 5.9(b)**, and the generated unique states are recorded in **Table 5.10**. Each unique state is considered to represent $\log_2 \lfloor 2d \rfloor = \beta$ bits of the external data. In **Table 5.10**, Q_1 encodes '000', Q_2 encodes '001', $\cdots$, and Q_8 encodes '111'. To increase the robustness of Proposed Method I against unauthorized decoding, the encoded bits can be randomly assigned to the states.

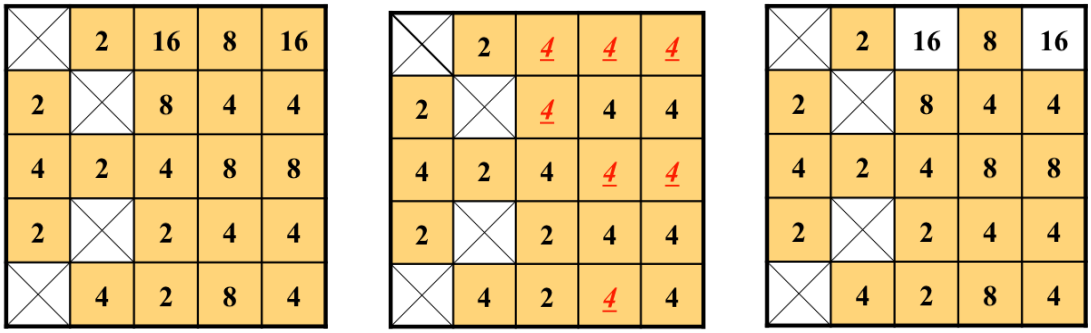
Based on the assigned bit sequence, the original ZRV are modified to the respective states depending on the external data to be embedded. By using **Table 5.10**, the ZRV pairs are left unmodified (i.e., $Q_1 = [O_1, O_2, O_3, O_4]$) if the external data is '000'. On the other hand, if the external data is '010', the ZRV pairs are modified to $Q_3 = [O_3, O_4, O_1, O_2]$, and so forth.

To extract the embedded data and recover the original image, the original state for every manipulated block $B'(m,n)$ must be identified as the reference state. The AC coefficients are grouped and unique states S'_γ are generated using **Equation (5.7)**. For this purpose, **Property 1** is exploited to locate the original state of each block by computing the product of run and magnitude $\Gamma(Q'_\gamma)$ for each Q'_γ as follows:

$$\theta(Q'_\gamma) = [(run_1 + run_2 + 2)(\mid val_{d-1} \mid + \mid val_d \mid)] - [(\mid val_1 \mid + \mid val_2 \mid)(run_{d-1} + run_d + 2)]. \tag{5.8}$$

Since it is possible that $run_1 + run_2 = 0$ or $run_{d-1} + run_d = 0$, $+2$ is introduced in the equation. The result for each state is collected. The original state always yields the largest $\theta(Q'_\Gamma)$ because in the original state, (a) the magnitude for the first group and the runs on the last group are of large values, and (b) the runs for the first group and the magnitude on the last group are of small values. Once the original state is identified, the manipulated block $B'(m,n)$ can be recovered to its original form $B(m,n)$ by reordering the ZRV pairs. The table associating states and bit sequences is referred to extract the embedded data. Decoding is performed on all blocks and finally, the recovered image G' is obtained.

However, not all the blocks satisfy **Property 1**. For example, in Lenna, $\sim$6% of the blocks failed the assumed property. To handle these failure blocks, we propose two



(a) Maximum number of groups (i.e., 2^{β}) in each block

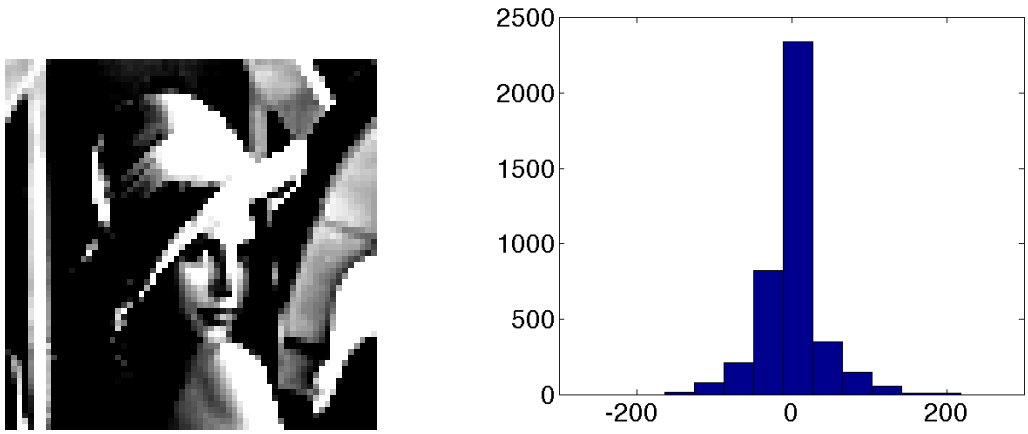
(b) Maximum number of groups utilized in each block when $a_{gLim} = 4$

(c) Blocks utilized when $a_{bLim} = 8$

Figure 5.10: Example of limiting the utilization of groups and blocks using a_{bLim} and a_{gLim}

solutions: one is to preprocess these blocks to fit the assumed property, while another is to utilize side information to indicate those failure blocks. Specifically, preprocessing removes the ZRV pair(s) from the back (with respect to zigzag order) until the assumed property is satisfied, thus making the proposed method irreversible (i.e., original compressed image cannot be recovered perfectly), but rewritable. Here, rewritable implies that G' can be re-utilized in the proposed method without repeating the preprocessing, or in other words, no further image quality degradation in the image for subsequent utilization (as discussed in **Section 2.4.1**). However, if reversibility is important for the application in question, the second option can be pursued. In particular, location of the failure blocks are recorded as side information and these blocks are avoided during the encoding and decoding processes. However, the embedding capacity is affected because: (1) failure blocks are not unutilized hence not contributing to the embedding capacity, and; (2) part of the capacity is utilized to store the side information. Hereinafter, in the case of rewritable, we called it as rewritable mode, while in the case of reversible, we called it as reversible mode.

Last but not least, the parameters a_{gLim} and a_{bLim} are introduced to control the quality and embedding capacity of the proposed method. The first parameter, a_{gLim}, limits the



(a) DC image (b) Distribution of difference in DC values

Figure 5.11: Sketch of Lenna image using the DC coefficients and distribution of the difference between neighboring DC coefficients

number of groups (i.e., setting the maximum value) that can be utilized for every block in the encoding process. For example, suppose image G consists of 5×5 blocks and the number of groups that can be formed in every block is as indicated in **Figure 5.10(a)**. In the same figure, the symbol $\boxtimes$ denotes a failure block. **Figure 5.10(b)** shades the blocks considered by Proposed Method I when $a_{gLim} = 4$ and indicates the actual number of groups formed in each shaded block. Note that although there are more available ZRV pairs that can be utilized (e.g., the block located at the 1-st row and 3-rd column in **Figure 5.10(a)** has 16 groups), at most $a_{gLim} = 4$ groups are considered. On the other hand, the second parameter, a_{bLim}, limits the number of blocks to be considered. Here, only the blocks with at most a_{bLim} groups of ZRV pairs are selected for encoding. For instance, when $a_{bLim} = 8$, only the blocks containing at most 8 groups (i.e., $4 \leq X \leq 31$) of ZRV pairs are utilized for scrambling-embedding. The result of imposing $a_{bLim} = 8$ to G is shown in **Figure 5.10(c)**, where only the shaded blocks are considered for encoding.

5.6.3 SE in DC Coefficients and its Natural Property (NIPC2)

By construction, the DC coefficient is the average intensity value for its corresponding 8×8 pixel block. Therefore, the DC coefficients themselves are sufficient to sketch the original image. An example is shown in **Figure 5.11(a)**, which suggests the leak-

age of image information (W. Li & Yuan, 2007). Therefore, it is crucial to scramble the DC coefficients to prevent unauthorized viewing. In this section, a permutation-based technique is proposed to scramble the DC coefficients while embedding data.

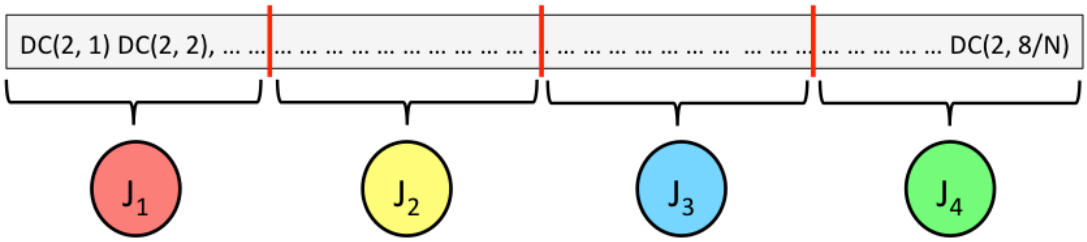
First, the DC coefficients are gathered into a matrix D, and let $D(i,j)$ denote the DC coefficient originating from the (i,j)-th block for $i \in \{1,2,\cdots,M\}$ and $j \in \{1,2,\cdots,N\}$. **Figure 5.11(b)** shows the distributions of $D(i,j) - D(i,j+1)$, i.e., horizontal differences, which follows the Laplacian distribution. The same trend is also observed for the vertical differences. Hence, **Property 2** holds true:

***Property 2**: DC coefficients are highly correlated in both the horizontal and vertical directions, i.e., $D(i,j) \sim D(i \pm \delta_j, i \pm \delta_i)$ for small δ_i and δ_j.*

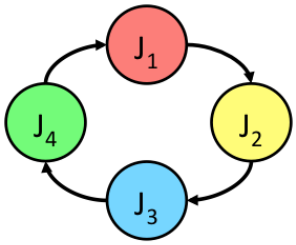
This property is exploited in both directions in Proposed Method II to achieve scrambling-embedding using the DC coefficients. First, the 1st row of D is left unmodified and the encoding proceeds from the 2nd row onwards. Each row of D is divided into non-overlapping groups (i.e., μ groups in total) and each group is denoted by J_f for $f \in \{1,2,\cdots,\mu\}$. Next, these groups are permuted to generate unique states Υ_ξ, where $\xi \in \{1,2,\cdots,\mu!\}$. Finally, each state is utilized to encode $\lfloor \log_2 \mu! \rfloor$ bits of the external data. In other words, each row of D is rearranged according to the external data. The same operations are applied in the vertical direction.

An illustration for $\mu = 4$ is shown in **Figure 5.12(a)**. **Table 5.11** records all possible states when $\mu = 4$ and their encoded bit sequences. For example, the row is modified from $\Upsilon_1 = [J_1, J_2, J_3, J_4]$ to $\Upsilon_{11} = [J_1, J_4, J_2, J_3]$ to embed '1010'. Note that the association between states and the bit sequences can be shuffled to increase the robustness of the proposed method.

Next, distortion is controlled by restricting: (a) the scope of permutation; and, (b)



(a) Division of DC coefficients into groups J_f



(b) Arranging DC groups in circular representation

Figure 5.12: Arrangement of DC coefficients groups

the number of states to consider. Note that a new (unique) state is generated by permuting the groups J_f. Within each generated state, the distance (in terms of modulo arithmetic) between every two consecutive groups (in terms of their original indices) must be smaller than the parameter Ψ, where $1 \leq \Psi \leq \mu - 1$. Otherwise, the state in question is ignored. For instance, **Figure 5.12(b)** shows a valid state for $\Psi = 1$ because the distance between every two consecutive groups, namely J_1 to J_2, J_2 to J_3 and J_3 to J_4, is $\max\{1,1,1\} = 1$. Note that the permutation reduces to the conventional cyclic shift process when $\Psi = 1$. Therefore, distortion can be controlled by tuning the parameter Ψ. **Table 5.11** records the distance of each consecutive pair (see 4-th column) in every state in the case of $\mu = 4$.

The output image quality can be further controlled by using the parameter φ when $\Psi = 1$, where φ is utilized to limit the number of states involved in the encoding process. If $\varphi = 2$, only Υ_1 and Υ_2 (or any two states achievable by $\Psi = 1$) are considered. Nonetheless, the utilization of Ψ and φ also reduces the embedding capacity because fewer states are considered. .

During the decoding process, each scrambled-embedded row is divided into μ groups

Table 5.11: List of all states and their corresponding assigned bit sequences

Υ_ξ	States	Encoded Bits	Distances	Ψ
Υ_1	$\{J_1, J_2, J_3, J_4\}$	0000	1, 1, 1	1
Υ_2	$\{J_4, J_1, J_2, J_3\}$	0001	1, 1, 1	1
Υ_3	$\{J_3, J_4, J_1, J_2\}$	0010	1, 1, 1	1
Υ_4	$\{J_2, J_3, J_4, J_1\}$	0011	1, 1, 1	1
Υ_5	$\{J_1, J_3, J_4, J_2\}$	0100	2, 1, 1	2
Υ_6	$\{J_4, J_2, J_3, J_1\}$	0101	2, 1, 2	2
Υ_7	$\{J_3, J_1, J_2, J_4\}$	0110	2, 1, 2	2
Υ_8	$\{J_2, J_4, J_1, J_3\}$	0111	2, 1, 2	2
Υ_9	$\{J_1, J_2, J_4, J_3\}$	1000	1, 2, 3	3
Υ_{10}	$\{J_1, J_3, J_2, J_4\}$	1001	2, 3, 2	3
Υ_{11}	$\{J_1, J_4, J_2, J_3\}$	1010	3, 2, 1	3
Υ_{12}	$\{J_1, J_4, J_3, J_2\}$	1011	3, 3, 3	3
Υ_{13}	$\{J_2, J_1, J_3, J_4\}$	1100	3, 2, 1	3
Υ_{14}	$\{J_2, J_1, J_4, J_3\}$	1101	3, 3, 3	3
Υ_{15}	$\{J_2, J_3, J_1, J_4\}$	1110	1, 2, 3	3
Υ_{16}	$\{J_2, J_4, J_3, J_1\}$	1111	2, 3, 2	3
Υ_{17}	$\{J_3, J_1, J_4, J_2\}$	-	2, 3, 2	3
Υ_{18}	$\{J_3, J_2, J_1, J_4\}$	-	3, 3, 3	3
Υ_{19}	$\{J_3, J_2, J_4, J_1\}$	-	3, 2, 1	3
Υ_{20}	$\{J_3, J_4, J_2, J_1\}$	-	1, 2, 3	3
Υ_{21}	$\{J_4, J_1, J_3, J_2\}$	-	1, 2, 3	3
Υ_{22}	$\{J_4, J_2, J_1, J_3\}$	-	2, 3, 2	3
Υ_{23}	$\{J_4, J_3, J_1, J_2\}$	-	3, 2, 1	3
Υ_{24}	$\{J_4, J_1, J_2, J_1\}$	-	1, 1, 3	3

and permuted exhaustively to generate all possible states Υ'_ξ. Here, the original state for every row is identified to enable image recovery and correct data extraction. For that purpose, the sum of differences $\Delta\varepsilon(\Upsilon'_\xi)$ is calculated using the DC coefficients in the current and previous rows for every possible state:

$$\Delta\varepsilon(\Upsilon'_\xi) = \sum_{j=1}^{N/8} \mid D_\xi(i,j) - D(i-1,j)\mid, \tag{5.9}$$

where $D_\xi(i,j)$ depends on the grouping of DC coefficients and the state ξ. The state with the smallest value of $\Delta\varepsilon(\Upsilon'_\xi)$ is declared as the original state (**Property 2**). Since the DC

coefficients are correlated vertically, the sum of difference between the original state and previous row should be the smallest. Hence, the DC coefficients can be restored to their original positions and the processes are repeated for all rows. Since the original state can be identified, the embedded data can also be extracted correctly.

Nonetheless, some rows in D (e.g., from similar texture area) fail to satisfy **Property 2** and hence they cannot be restored perfectly. These unfit rows are identified using **Equation (5.11)** in the preprocessing step and their indices are recorded as side information. Here, they are merely permuted without embedding data. Similar steps are performed for all columns.

5.6.4 SE in Total Block Energy and its Natural Property (NIPC3)

The sum of magnitude of quantized AC coefficients $\Lambda(i,j)$ from the (i,j)-th 8×8 DCT block is calculated as follows:

$$\Lambda(i,j) = \left(\sum_{u=1}^{8} \sum_{v=1}^{8} | B(i,j)_{u,v} | \right) - |B(i,j)_{1,1}|. \tag{5.10}$$

Due to the nature of DCT, $\Lambda(i,j)$ yields a large value for a complex or edge block, and vice versa. Leakage of these information may allow an intruder to derive additional image information of the scrambled image (Minemura et al., 2012). Thus, it is crucial to permute the AC coefficient blocks to avoid unauthorized viewing. **Figure 5.13** shows the distribution of $\Lambda(i,j) - \Lambda(i,j-1)$. The distribution indicates that neighboring blocks have similar energies, and hence they are highly correlated. Similar trend is also observed for the vertical direction, i.e., $\Lambda(i,j) - \Lambda(i-1,j)$. In particular, the following property in JPEG is considered for the rest of this section:

Property 3*: The sum of magnitude for quantized AC coefficient blocks are highly correlated in both the horizontal and vertical directions.*

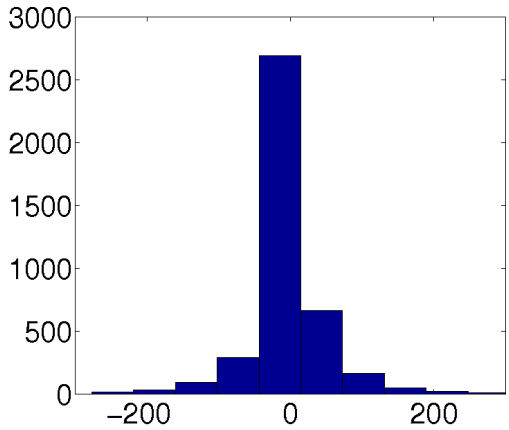


Figure 5.13: Distribution for difference in AC Block Energies in Lenna

Here, the permutation function detailed in the previous section is applied on Λ to realize scrambling-embedding. Again, preprocessing is first performed to identify the unfit rows (of AC energy blocks) by calculating the sum of differences between the current and previous rows. Then, location of the unfit rows are identified and recorded as side information. Each of the rows that fit the assumed property is divided into groups to generate all possible states. These possible states are then utilized to encode data, where each row in Λ is modified based on the external data to be embedded, similar to the case of NIPC2 (DC coefficient).

For decoding, the sum of differences for every possible states is calculated. The state which yields the smallest sum of differences is the original state. Therefore, the blocks are recovered to their original position and the embedded data is obtained. The encoding and decoding processes here are similar to those of NIPC2 and the details are omitted here.

5.6.5 Combined Methods

NIPC1, NIPC2 and NIPC3 are operating independently on non-overlapping areas in a JPEG compressed image. Therefore, they can be combined to increase the range of image quality degradation by scrambling while increasing the embedding capacity by

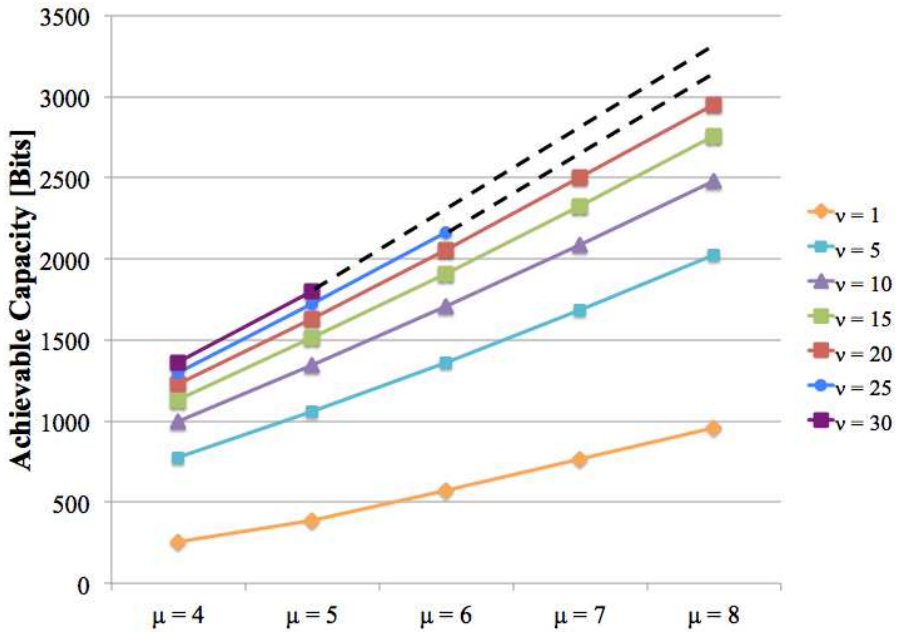


Figure 5.14: Possible embedding capacity for combined rows for an image of 512×512 pixels. The dotted line indicates the predicted possible capacity because the number of permutations is too huge to be calculated.

embedding. The performance of the combined method in terms of achievable embedding capacity, output image quality degradation and bitstream size increment are presented in **Section 5.8**.

5.7 NIPC: Discussions

This section discusses the improvement of NIPC on capacity and robustness (**Section 5.7.1** and robustness of NIPC against well-suited attacks (**Section 5.7.2**).

5.7.1 Improvement on Capacity and Robustness

The embedding capacity for NIPC2 and NIPC3 can be further enhanced by increasing the number of possible permutations. **Figure 5.14** shows the increment of the highest possible capacity when the number of groups (i.e., μ) in a row is increased. In addition, it is also observed that the highest possible capacity increases if two or more rows are combined. For instance, the highest possible capacity for a single row (i.e., $\nu = 1$) is 256 bits for an 512×512 image and $\mu = 4$. After increasing ν to 5, the achievable capacity is 776 bits, which is $\sim$3 times larger than that of $\nu = 1$. Here, the possible capacity for

every ν and μ is calculated by using the following equation:

$$\lfloor M/\nu \rfloor \times \lfloor \log_2(\nu\mu!) \rfloor + \lfloor \log_2[(((M) \mod \nu) \times \mu)!] \rfloor. \tag{5.11}$$

Therefore, the capacity increases when μ increases or when rows are combined.

5.7.2 Robustness against Well-suited Attacks

This subsection discusses the robustness of the proposed method if **Property 1, 2,** and **3** of the proposed method are known and exploited by the attacker.

First, for NIPC1, the attacker can perform the reverse operation using **Equation (5.8)**. However, the AC coefficient blocks are scrambled using NIPC2. In other words, the attacker can only recover the original order of the ZRV pairs within individual blocks but does not know the exact location of each block. Hence, the original image or its sketch remains concealed. Even if the attacker knew **Properties 2** and **3**, he still needs to guess the parameter values μ and ν correctly for NIPC2 and NIPC3, respectively, in order to successfully perform the reverse operations.

Secondly, the tables utilized to encode the external data are kept private between the sender and receiver. Therefore, even in the event where the attacker successfully recovers the image (fully or partially), he is still not able to extract the embedded external data.

Next, substitution can be performed on ZRV pairs and DC coefficients after performing the scrambling-embedding processes to further improve robustness. Specifically, the values of the original ZRV pairs and DC coefficients are substitute by another value in the same category. Note that these operations will not cause any bitstream size increment because the substitution is done within the same category. For the case of ZRV pairs, the Huffman codewords of exact same length can be bijectively mapped. For example, since both Run/Size of 0/3 (i.e., codeword = 100$\mathscr{X}\mathscr{Y}\mathscr{Z}$) and 2/1 (i.e., codeword = 11100$\mathscr{W}$) have the same complete Huffman codeword length of 6 bits, they can be bijec-

tively mapped without giving any impact to the bitstream size. In fact, more codewords satisfying the length condition can be considered. Without knowing the substitution or mapping rules, the attacker is not able to reverse the operations.

In addition, the number of groups per row μ and the number of row ν to be grouped can be changed periodically. Also, instead of evenly dividing the DC coefficients into groups, irregular division can be performed. Similar modification can be done to the AC coefficient blocks.

Hence, even if the attacker exploits **Property 1, 2** and **3**, the aforementioned discussions suggest that the reverse operations are not straight forward for an attacker who knows nothing about the parameter values. Therefore, the proposed method is robust against well-suited attacks.

5.8 NIPC: Experimental Results

Experiments were conducted to verify the performance of each proposed technique using the UCID (Uncompressed Color Image Database) database (Schaefer & Stich, 2004). The images in UCID database are all of size 384×512 or 512×384 pixels. These images are converted into grayscale and compressed at quality factor 80 for experiment purposes. Change on bitstream size, effective embedding capacity, and output image (i.e., scrambled-embedded) quality for NIPC1, NIPC2 and NIPC3 are examined in the following subsections. In addition, performance of the integrated method (i.e., NIPC1, NIPC2 and NIPC3) is evaluated. Finally, comparisons of the proposed method with related conventional methods are also performed and discussed.

5.8.1 Image Quality Degradation, Bitstream Size and Carrier Capacity

Table 5.12 shows the average performance of NIPC1 using the UCID database for various parameters in rewritable mode (i.e., preprocessing is applied). Note that, by de-

Table 5.12: Performance of NIPC1 (ZRV Pairs) using various parameters in rewritable mode and reversible mode (the right-most column)

Mode	Rewritable							Reversible
Parameter	a_{gLim}	a_{gLim}	a_{gLim}	a_{bLim}	a_{bLim}	a_{bLim}	$a_{\text{gLim}} = a_{\text{bLim}}$	$a_{\text{gLim}} = a_{\text{bLim}}$
Value	$= 2$	$= 4$	$= 8$	$= 2$	$= 4$	$= 8$	$= 16$	$= 16$
Image Quality Degradation								
G (SSIM)	0.9640							
G (PSNR)	37.07							
G' (SSIM)	0.5580	0.4365	0.3904	0.9432	0.8040	0.4736	0.3848	0.4117
G' (PSNR)	15.97	14.69	14.05	34.19	25.11	15.61	13.97	14.15
Bitstream Size Changes								
G' [%]	-0.65	-0.66	-0.66	-0.66	-0.66	-0.66	-0.66	0.00
Effective Capacity								
G' [Bits]	4991.84	7149.68	8578.64	676.15	2862.82	7342.12	8887.77	5137.65
G' [BpNAC]	0.11	0.16	0.19	0.01	0.06	0.16	0.19	0.12

$*$ G and G' are the original and output images, respectively.

$*$ Effective capacity is obtained after deducting the raw capacity with location map of size $M \times N$.

sign, only blocks which contain at least 2 groups of ZRV pairs (i.e., at least 4 ZRV pairs) are utilized in the experiment.

The achieved image quality (a measure for distortion) in terms of SSIM ranges from 0.3848 to 0.9432, and PSNR ranges from 13.97 to 34.19 dB. In terms of effective capacity, NIPC1 offers up to 8887.77 bits (equivalent to 0.19 bits per non-zero AC coefficient [BpNAC]) using all the blocks in rewritable mode. The bitstream size increment is -0.66%, on average, for all considered parameters in rewritable mode. Here, some gain in compression is achieved due to preprocessing that removes ZRV pairs from the block. In general, it is also observed that the image quality is progressively degraded and the effective capacity is progressively increased when either a_{gLim} or a_{bLim} increases.

For completion of discussion, the performance of NIPC1 in reversible mode is also examined and the results are recorded in **Table 5.12**. On average, 7.91% of the blocks are only scrambled (i.e., without data embedding) because these blocks failed to sat-

Table 5.13: Performance of NIPC2 (DC Coefficients) for various μ and Ψ

μ	4	5	6	7	8						
Ψ	Max	Max	Max	Max	1	2	3	4	5	6	Max
Image Quality Degradation											
G (SSIM)	0.9640										
G (PSNR)	37.07										
G'_{R_o} (SSIM)	0.5845	0.5912	0.5744	0.5506	0.5542	0.5507	0.5486	0.5473	0.5466	0.5461	0.5464
G'_{R_o} (PSNR)	13.52	13.72	13.49	13.22	13.15	13.13	13.12	13.12	13.12	13.12	13.13
$G'_{R_o \oplus C_o}$ (SSIM)	0.3813	0.3956	0.3740	0.3571	0.3596	0.3584	0.3569	0.3560	0.3552	0.3551	0.3553
$G'_{R_o \oplus C_o}$ (PSNR)	10.54	10.84	10.55	10.41	10.38	10.37	10.37	10.37	10.37	10.37	10.38
Bitstream Size Changes											
G'_{R_o} [%]	0.14	0.26	0.30	0.38	0.04	0.18	0.27	0.31	0.34	0.36	0.37
$G'_{R_o \oplus C_o}$ [%]	3.95	3.79	4.15	4.39	4.41	4.42	4.43	4.44	4.44	4.45	4.44
Effective Capacity											
G'_{R_o} [Bits]	153.16	247.13	383.65	493.65	99.00	186.32	307.30	369.29	379.20	338.65	302.15
G'_{R_o} [BpDC]	0.05	0.08	0.12	0.16	0.03	0.06	0.10	0.12	0.13	0.11	0.10
G'_{C_o} [Bits]	172.68	286.48	457.52	625.71	115.16	228.02	387.41	479.50	558.04	584.05	644.70
G'_{C_o} [BpDC]	0.06	0.09	0.15	0.20	0.04	0.08	0.13	0.16	0.19	0.20	0.21
$G'_{R_o \oplus C_o}$ [Bits]	325.84	533.61	841.16	1119.35	214.16	414.34	694.72	848.79	937.24	922.70	946.85
$G'_{R_o \oplus C_o}$ [BpDC]	0.11	0.17	0.27	0.36	0.07	0.14	0.23	0.29	0.32	0.31	0.31

∗ G is the original image.

∗ G'_{R_o} is the output image for row operation.

∗ G'_{C_o} is the output image for column operation.

∗ $G'_{R_o \oplus C_o}$ is the output image for row and column operations.

∗ Effective Capacity is obtained after deducting the side information which records the location of unfit row(s) and column(s) (i.e., $\lceil \log_2 M \rceil$ bits per unfit row and $\lceil \log_2 N \rceil$ bits per unfit column).

isfy **Property 1**. In this case, a location map is needed to differentiate the scrambled-embedded blocks from the scrambled-only blocks. Although the raw capacity for NIPC1 in reversible mode is 8209.65 bits, the effective capacity is 5137.65 bits after storing the location map as side information. Nevertheless, bitstream size remains unchanged (up to byte alignment level) and there is no loss of information in the output image.

Similarly, **Table 5.13** records the average results of the output image quality, changes

Table 5.14: Performance of NIPC2 (DC Coefficients) using various φ when $\mu = 4$ and $\Psi = 1$

φ	2	3	4
	Image Quality Degradation		
G'_{R_o} (SSIM)	0.7048	0.6197	0.5871
G'_{R_o} (PSNR)	15.57	13.96	13.54
$G'_{R_o \oplus C_o}$ (SSIM)	0.5164	0.4145	0.3833
$G'_{R_o \oplus C_o}$ (PSNR)	12.13	10.78	10.54
	Bitstream Size Changes		
G'_{R_o} [%]	0.02	0.03	0.04
$G'_{R_o \oplus C_o}$ [%]	2.46	3.46	3.93
	Effective Capacity		
G'_{R_o} [Bits]	47.95	46.41	96.00
G'_{R_o} [BpDC]	0.02	0.02	0.03
G'_{C_o} [Bits]	57.28	57.46	115.12
G'_{C_o} [BpDC]	0.02	0.02	0.04
$G'_{R_o \oplus C_o}$ [Bits]	105.23	103.86	211.13
$G'_{R_o \oplus C_o}$ [BpDC]	0.03	0.03	0.07

on bitstream size, and effective capacity achieved by NIPC2 using the UCID database. By using different parameters and types of operation (i.e., row and/or column operation), the achieved image quality in terms of SSIM ranges from 0.3551 to 0.5912, and PSNR ranges from 10.37 to 13.72 dB. The average effective capacity achieved by NIPC2 ranges from from 99.00 bits (0.03 bits per DC coefficient [BpDC]) to 1119.35 bits (0.36 BpDC). Overall, it is observed that the effective capacity increases when either μ or Ψ increases for both the row and column operations. In addition, the increment of bitstream size in the output image after the row operation ranges from 0.04% to 0.38% while the increment after both row and column operations ranges from 3.79% to 4.45%. The bitstream size increment due to row operation is comparatively low because these operations destroy the vertical correlation of the DC coefficients, which is irrelevant to DPCM in JPEG. However, column operations destroy the horizontal correlation of the DC coefficients,

Table 5.15: Performance of NIPC3 (AC Block Energy) for various μ and Ψ

μ	4	5	6	7	8						
Ψ	Max	Max	Max	Max	1	2	3	4	5	6	Max
Image Quality Degradation											
G (SSIM)	0.964										
G (PSNR)	37.07										
G'_{R_o} (SSIM)	0.4775	0.4722	0.4471	0.4137	0.4075	0.4057	0.4058	0.4055	0.4057	0.4054	0.4052
G'_{R_o} (PSNR)	19.58	19.51	19.27	19.02	18.93	18.92	18.92	18.92	18.93	18.92	18.92
$G'_{R_o \oplus C_o}$ (SSIM)	0.3299	0.3315	0.3143	0.3014	0.3002	0.2995	0.2995	0.2994	0.2992	0.2991	0.2991
$G'_{R_o \oplus C_o}$ (PSNR)	18.58	18.59	18.46	18.39	18.37	18.36	18.36	18.36	18.36	18.36	18.36
Effective Capacity											
G'_{R_o} [Bits]	99.46	117.55	135.84	98.21	87.26	144.99	213.51	232.96	195.25	139.43	95.70
G'_{R_o} [BpB]	0.03	0.04	0.04	0.03	0.03	0.05	0.07	0.08	0.06	0.05	0.03
G'_{C_o} [Bits]	152.98	210.75	293.48	259.96	104.20	191.09	295.89	334.32	306.50	234.82	318.19
G'_{C_o} [BpB]	0.05	0.07	0.10	0.08	0.03	0.06	0.10	0.11	0.10	0.08	0.10
$G'_{R_o \oplus C_o}$ [Bits]	252.44	328.30	429.32	358.17	191.46	336.08	509.41	567.28	501.75	374.25	418.89
$G'_{R_o \oplus C_o}$ [BpB]	0.08	0.11	0.14	0.12	0.06	0.11	0.17	0.18	0.16	0.12	0.13

$*$ G is the original image.

$*$ G'_{R_o} is the output image for row operation.

$*$ G'_{C_o} is the output image for column operation.

$*$ $G'_{R_o \oplus C_o}$ is the output image for row and column operations.

$*$ Effective Capacity is obtained after deducting the side information which records the location of unfit row(s) and column(s) (i.e., $\lceil \log_2 M \rceil$ bits per unfit row and $\lceil \log_2 N \rceil$ bits per unfit column).

making DPCM ineffective and hence leading to greater bitstream size increment. To further control the degradation in image quality, parameter φ can be utilized and the results for various φ for $\mu = 4$ and $\Psi = 1$ are shown in **Table 5.14**. When φ increases, the output image quality is progressively degraded due to the row and column operations, where SSIM ranges from 0.3833 to 0.7048 and PSNR ranges from 10.54 to 15.57 dB. Similarly, the effective capacity is progressively increased when φ increases, with bitstream size increment of less than 4%.

Next, **Table 5.15** shows the average results for NIPC3 obtained by using the AC block energy Λ. Again, the performance is measured in terms of output image quality,

Table 5.16: Performance of NIPC3 (AC Block Energy) using various φ when $\mu = 4$ and $\Psi = 1$

φ	2	3	4
	Image Quality Degradation		
G'_{R_o} (SSIM)	0.6353	0.5275	0.4777
G'_{R_o} (PSNR)	21.35	20.09	19.60
$G'_{R_o \oplus C_o}$ (SSIM)	0.4612	0.3627	0.3302
$G'_{R_o \oplus C_o}$ (PSNR)	19.56	18.82	18.59
	Effective Capacity		
G'_{R_o} [Bits]	37.95	31.84	75.43
G'_{R_o} [BpB]	0.01	0.01	0.02
G'_{C_o} [Bits]	54.75	55.29	112.93
G'_{C_o} [BpB]	0.02	0.02	0.04
$G'_{R_o \oplus C_o}$ [Bits]	92.69	87.13	188.36
$G'_{R_o \oplus C_o}$ [BpB]	0.03	0.03	0.06

† No changes on bitstream size up to 2 decimal places for G'_{R_o} and $G'_{R_o \oplus C_o}$.

change in bitstream size, and effective capacity for different parameters using the UCID database. Similar to NIPC2, NIPC3 is also able to achieve different level of distortion and effective capacity. The output image quality ranges from 0.2991 to 0.4775 in terms of SSIM, and ranges from 18.36 dB to 19.58 dB in terms of PSNR. The effective capacity ranges from 87.26 bits (0.03 bits per AC Block [BpB]) to 567.28 bits (0.18 BpB). Similar to the results achieved by NIPC2, the image quality is progressively degraded and the effective capacity is progressively increased when μ and Ψ increase. Nevertheless, output images for NIPC3 do not cost noticeable change on bitstream size ($\sim$0%) for both row and column operations. Unlike NIPC2 which is implemented to process the DC coefficients, destroying the horizontal and vertical correlations in AC Block Energy level do not affect the bitstream size. This is because the AC coefficients are coded independently within an 8×8 coefficients block in JPEG and hence permuting the AC blocks does not affect the coding efficiency. Similarly, the perceptual quality is controllable in NIPC3

Table 5.17: Performance of the proposed combined method

	NIPC1[1] $\oplus$ NIPC2[2] $\oplus$ NIPC3[2]
Image Quality Degradation (SSIM)	0.0548
Image Quality Degradation (PSNR)	8.63
Bitstream Size Changes [%]	4.48
Effective Capacity [Bits]	10238.01
Effective Capacity [Bits Per Pixels]	0.05

[1] uses $a_{gLim} = a_{bLim} = 16$.

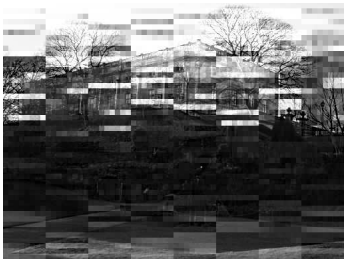
[2] uses $\mu = 8$ and $\Psi = \text{Max}$.

using φ, and the results are recorded in **Table 5.16**. Here, the output image quality is progressively degraded when φ increases. In particular, the achievable image quality ranges from 0.3302 to 0.6353 in terms of SSIM, and ranges from 18.59 to 21.35 dB in terms of PSNR. In addition, the achievable effective capacity ranges from 31.84 to 188.36 bits when various φ are considered. Here, the effective capacity increases when φ increases, with insignificant change on bitstream size.

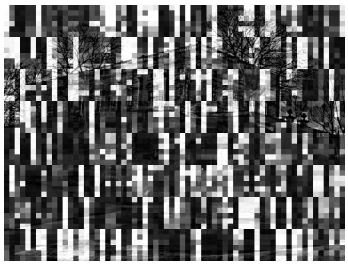
The output images for NIPC1, NIPC2 and NIPC3, as well as the combined method (i.e., NIPC1, NIPC2 and NIPC) are shown in **Figure 5.15**. The average results for the combined method are recorded in **Table 5.17**. As the system of combined methods, the lowest image quality of 0.0548 (i.e., SSIM) or 8.63 dB (i.e., PSNR) is achieved by invoking all NIPC1, NIPC2 and NIPC3. On the other hand, the highest image quality of SSIM = 0.9432 (or PSNR = 34.19dB) is achieved by using NIPC3 only. From the results, it is found that the bitstream size increment is dominated by NIPC1, and thus the combined methods achieves similar bitstream size increment as in NIPC1, which is $\sim$4.48%. As a system, the combined method offers, on average, 10238.01 bits (0.05 bits per pixel), as the highest effective capacity when using the UCID database.

(a) Original Image

(b) NIPC1
($a_{gLim} = 16$)



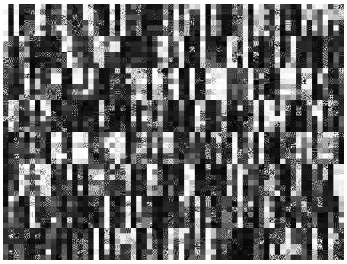
(c) NIPC2 - G'_{R_o}
($\mu = 8$ & $\Psi =$ Max)



(d) NIPC2 - $G'_{R_o \oplus C_o}$
($\mu = 8$ & $\Psi =$ Max)

(e) NIPC3 - G'_{R_o}
($\mu = 8$ & $\Psi =$ Max)

(f) NIPC3 - $G'_{R_o \oplus C_o}$
($\mu = 8$ & $\Psi =$ Max)

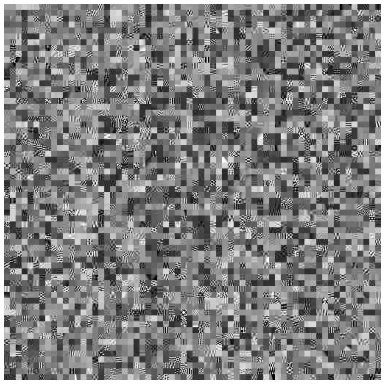


(g) Proposed Combined Method

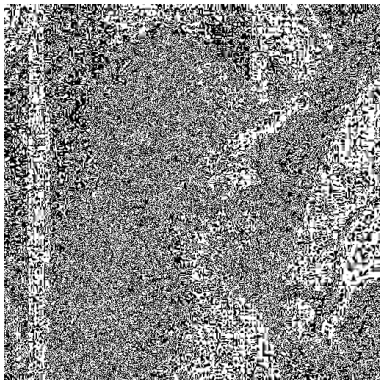
Figure 5.15: Output images from NIPC1, NIPC2, NIPC3 and the proposed combined method

5.8.2 Comparison with Related Works

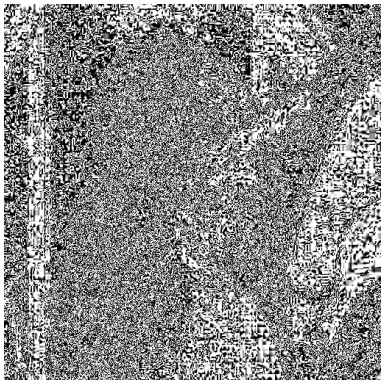
In this section, the proposed combined method (i.e., system) is compared with the relevant methods considered using the standard test images (i.e., Airplane, Baboon, Boat, Lake, Lenna, and Peppers (*The USC-SIPI Image Database [online]*, n.d.)), each compressed at the quality factor of 80. Zhang et al.'s (X. Zhang, 2012) and Ma et al.'s (Ma et al., 2013) methods are re-implemented to handle JPEG compressed image. Kundur et



(a) Proposed combined method

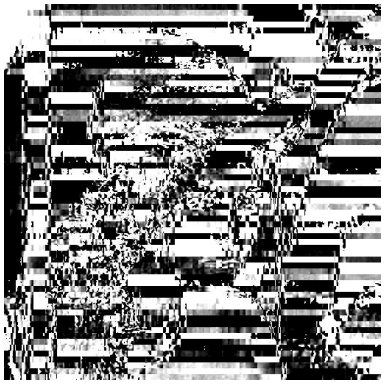


(b) Zhang et al.'s method (X. Zhang, 2012)



(c) Ma et al.'s method (Ma et al., 2013)

(d) Kundur et al.'s method (Kundur & Karthik, 2004)



(e) Qian et al.'s method (Qian et al., 2014)

Figure 5.16: Output image for the proposed combined method and related conventional methods.

al.'s method (Kundur & Karthik, 2004) and Qian et al.'s method (Qian et al., 2014) are also utilized as the benchmark methods in this study.

Table 5.19 shows the results for each considered method and the proposed method. Zhang et al.'s method and Ma et al.'s method are able to generate a significantly distorted output image, but the bitstream size increment is very large ($\sim$212%). On the other hand, the bitstream size increment for the proposed combined method is only $\sim$3.19%. Qian

Table 5.18: Comparative results for all sketch attacks

	DCM	NCC	EAC	PLZ
Zhang et al.'s method (X. Zhang, 2012)	Pass	Fail	Pass	Fail
Ma et al.'s method (Ma et al., 2013)	Pass	Fail	Pass	Fail
Kundur et al.'s method (Kundur & Karthik, 2004)	N/A	Fail	Fail	Fail
Qian et al.'s method (Qian et al., 2014)	Fail	Fail	Fail	Fail
Niu et al.'s method (Niu, Zhou, Ding, & Yang, 2008)	Fail	Pass	Pass	Pass
Proposed Method	Pass	Pass	Pass	Pass

et al.'s method is able to generate distorted image without bitstream size increment, but it's capacity is relatively low and it is irreversible. It is worth mentioning that the distortion achieved by the proposed method can be further intensified by randomizing the sign of each nonzero AC coefficient, which does not affect the bitstream size. Although Kundur et al.'s method achieves very high capacity and maintains the bitstream size, it is irreversible. In addition, all the conventional methods considered cannot progressively degrade the image quality or increase embedding capacity, but the proposed method is able to achieve a wide range of distortion and embedding capacity by adjusting the parameters a_{gLim}, a_{bLim}, μ, Ψ and φ. Therefore, the proposed combined method: (a) provides more flexibilities to suit the needs of various applications, (b) offers comparable performance when compared to the related conventional methods, and (c) suppresses bitstream size increment.

For visual comparison, **Figure 5.16** shows the output image for each method considered. It is observed that the output image for Zhang et al.'s, Ma et al.'s and Qian et al.'s methods are fairly distorted, and thus the perceptual meaning of the original image is somehow concealed. Kundur et al.'s method only partially distorts the image because the sign information is merely randomized. On the other hand, the distortion level of the output image can be controlled by the proposed combined method, and the most distorted output image resembles noise (i.e., block-based-like noise).

Table 5.19: Comparison with related conventional methods

	Image Quality (SSIM)	Image Quality (PSNR)	Carrier Capacity [Bits]	Bitstream Size Changes [%]	Reversibility	Scalability
Zhang et al.'s method [1]	0.0107	6.45	4300.80	212.50	Irreversible	No
Ma et al.'s method [2]	0.0108	6.45	32448.83	212.63	Reversible	No
Kundur et al.'s method [3]	0.4165	18.87	63698.17	-0.01	Irreversible	No
Qian et al.'s method [4]	0.0600	6.37	2040.00	0.00	Irreversible	No
Proposed Method	0.0638 - 0.9152	10.05 - 36.47	669.83 - 14861.67	-0.48 - 3.19	Reversible / Rewritable	Yes

[1] Zhang et al's method (X. Zhang, 2012)

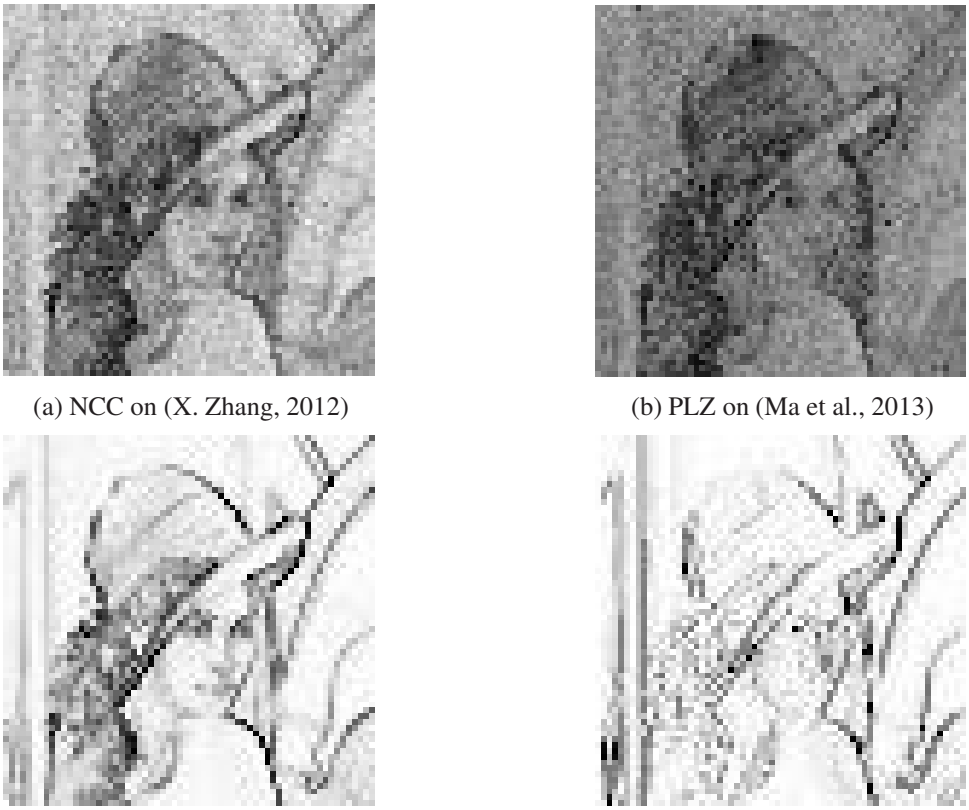
[2] Ma et al.'s method (Ma et al., 2013)

[3] Kundur et al.'s method (Kundur & Karthik, 2004)

[4] Qian et al.'s method (Qian et al., 2014)

Last but not least, four sketching techniques are considered to examine the robustness of the proposed and conventional methods considered. The first technique is known as DCM (DC category mapping) attack, where it assigns a representative value to each DC category (Ong et al., 2013). This technique is effective in attacking method which permutes the DC coefficients within the same category (e,g., Niu et al. method (Niu et al., 2008)). In addition, three techniques proposed in (Minemura et al., 2012), namely, NCC (non-zero AC coefficient count), EAC (energy of AC coefficients block), and PLZ (position of last non-zero AC coefficients), are considered to sketch the plaintext directly by using the AC components of the scrambled image. Detailed information about these attacks can be found in the **APPENDIX B**.

Zhang et al.'s method, Ma et al.'s method, Kundur et al.'s method, Niu et al.'s method (Niu et al., 2008), Qian et al.'s method (Qian et al., 2014), the integrated method $\oplus$, and the proposed combined method are compared in terms of robustness against the aforementioned sketch attacks. **Table 5.18** summarizes the outcome of the sketch attacks, which are based on visual inspection to determine whether the outline of the image is revealed. If the outline of the image is visible under a particular sketch attack (see



(a) NCC on (X. Zhang, 2012)

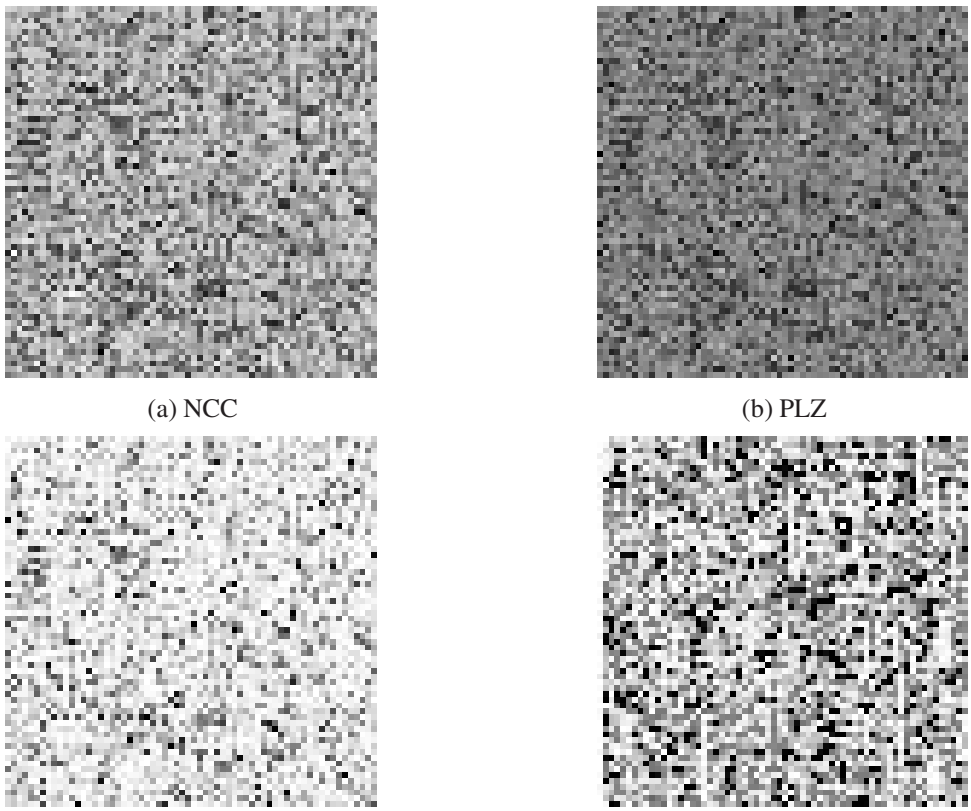
(b) PLZ on (Ma et al., 2013)

(c) EAC on (Kundur & Karthik, 2004)

(d) DCM on (Niu et al., 2008)

Figure 5.17: Sketches produced by the simple sketch attacks when applied on the related conventional methods. Here, only the selected successful cases are shown. Other outcomes are summarize in **Table 5.18**.

Figure 5.17 for successful attacks), it will be labeled as Fail in the table. Otherwise, it will be labeled as Pass (see **Figure 5.18** for failed attacks) because the image information remains concealed. Zhang et al.'s method and Ma et al.'s method pass the DCM and EAC attacks because the DC and AC coefficient values are modified by using the XOR operations. However, the attacker is able to sketch the outline of the plaintext image by using either NCC or PLZ attack because the number of non-zero AC coefficients and the position of the last non-zero AC coefficients remain intact in these methods. On the other hand, Kundur et al.'s method is vulnerable to all sketch attacks because only the sign information is randomized. Niu et al.'s method survives the NCC, PLZ and EAC attacks because it permutes the AC coefficient blocks. However, it fails the DCM attack. The sketch generated for these methods are shown in **Figure 5.17**. From these sketches, the



(a) NCC (b) PLZ

(c) EAC (d) DCM

Figure 5.18: Output of all four sketch attacks applied on the proposed combined method (NIPC1, NIPC2 and NIPC3).

outline of the Lenna image can be recognized. It should be noted that sketch of higher quality is achievable when the image resolution increases. Although Qian et al's method is able to conceal the perceptual meaning of the image, it is vulnerable to NCC and PLZ because the number of non-zero coefficient and the position of the last AC coefficient in each block remains unchanged, and its degree of distortion is not scalable. Similar to Niu et al.'s method, Qian et al.'s method also failed in DCM attack because it is operated using category mapping in DC coefficients. On the other hand, the proposed combined method and the integrated method $\oplus$ survive all four sketch attacks considered. The corresponding sketches for the proposed combined method are shown in **Figure 5.18**. Although $\oplus$ is able to survive all sketch attacks, its degree of distortion is not scalable and the bitstream size increment is > 2 times larger than that of the proposed combined method while embedding a smaller amount of data.

5.9 NIPC: Summary

A reversible unified information hiding method was proposed to realize scrambling-embedding in JPEG compressed image using S2E approach. Two techniques were proposed and applied to three non-overlapping components in JPEG compressed image, namely, the DC coefficients, AC block energy and zero-run value pairs. The proposed combined method could achieve scalability in embedding capacity, which ranged from 32 bits to 10238 bits, and scalability in perceptual quality degradation with SSIM value ranging from 0.0548 to 0.9432. In addition, the maximum observed bitstream size increment is 4.48%. Experimental results verified that the proposed combined method is able to achieve comparable performance with respect to the related conventional methods. The proposed combined method also survives all four sketch attacks while the related conventional methods fail for at least one sketch attack.

CHAPTER 6

PROPERTIES IN THE PROPOSED UIH METHODS: SCALABILITY, REVERSIBILITY AND COMMUTATIVE

6.1 Introduction

This chapter discusses the overall performance of each proposed method in **Chapter 3**, **Chapter 4** and **Chapter 5**, with respect to the important properties in UIH. The properties in UIH discussed in this section include (a) Scalability, (b) Reversibility, and (c) Commutative, and they are detailed in **Section 6.2**, **Section 6.3** and **Section 6.4**, respectively. **Section 6.5** discusses the limitation of the study, and **Section 6.6** summarizes the chapter.

6.2 Scalability

Scalability refers to the progression in image quality degradation and the ability to offer more carrier capacity when necessary. All the proposed methods aims at achieving both scalabilities.

Conventional external data insertion methods aim at achieving high output image quality, therefore, the achievable embedding capacity is limited. In unified information hiding, it aims to achieve external data insertion and scrambling at the same time. Thus, UIH relaxes the high image quality requirement and thus it is expected to achieve higher embedding capacity than the conventional methods.

6.2.1 Scalability in Image Quality Degradation

Figure 6.1 shows the average lowest SSIM, highest SSIM and SSIM range (i.e., the highest SSIM subtract with the lowest SSIM) for the methods proposed in this thesis, including (i) HAM proposed in **Chapter 3**, (ii) SURIH proposed in **Chapter 4**, (iii) NIPS

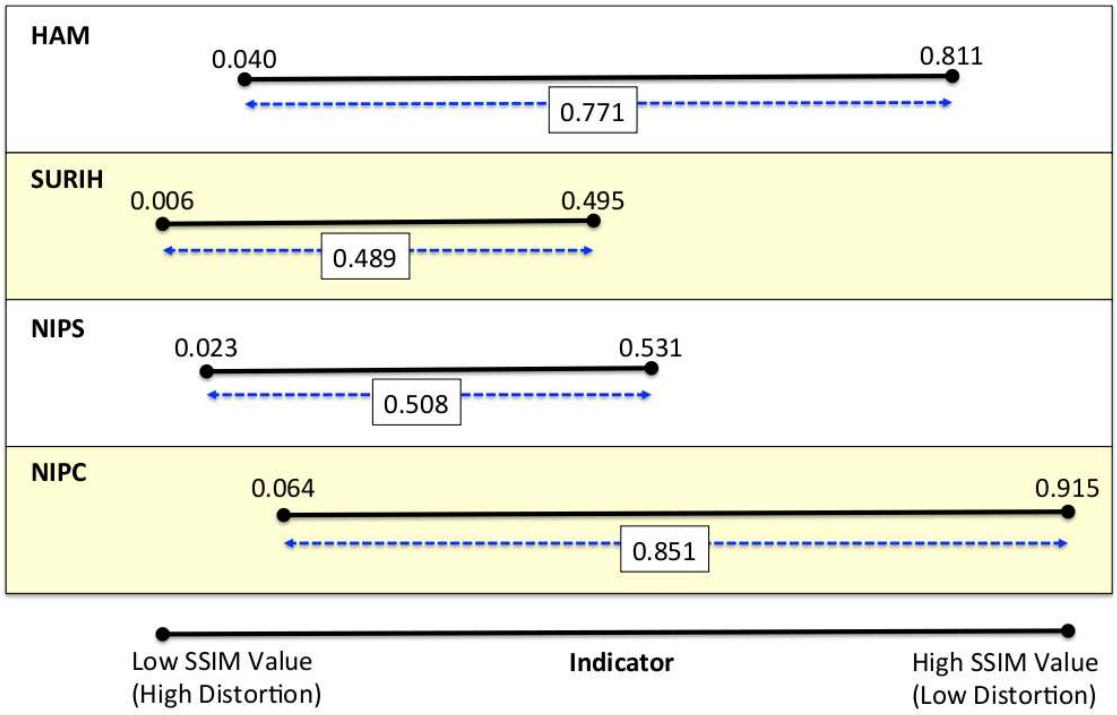


Figure 6.1: Comparison of Achievable Quality Degradation (SSIM) among the Four Proposed Methods

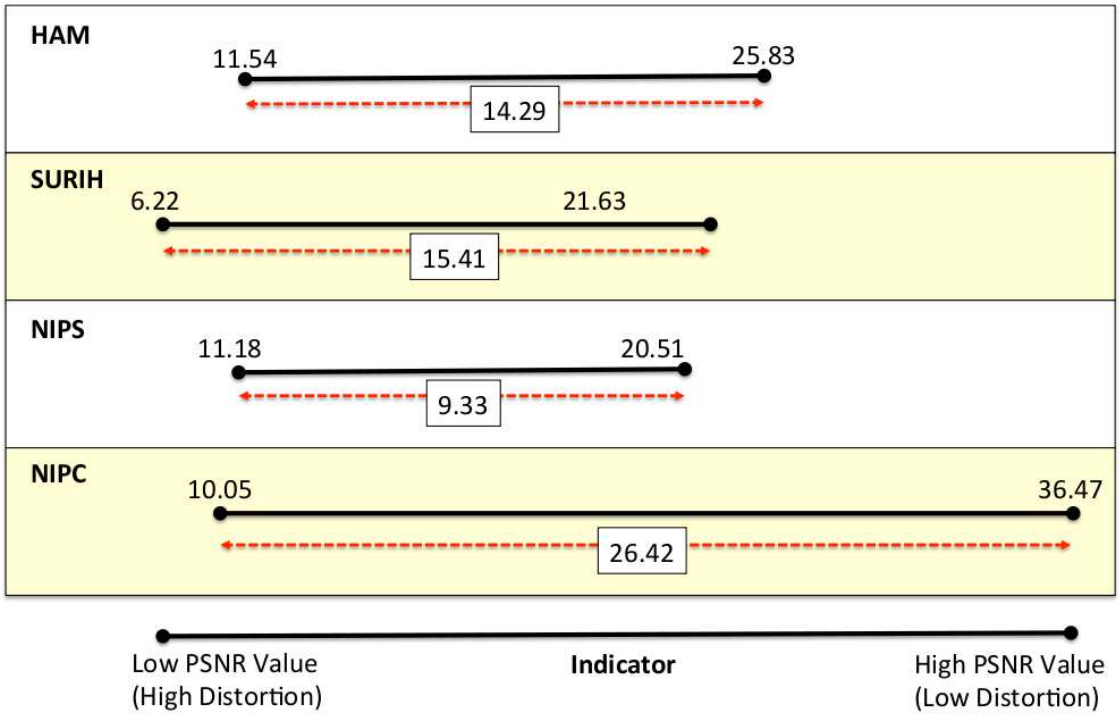


Figure 6.2: Comparison of Achievable Quality Degradation (PSNR) among the Four Proposed Methods

and (iv) NIPC proposed in **Chapter 5**. The results are obtained by using six 512×512 standard test images (i.e., Airplane, Baboon, Boat, Lake, Lenna and Peppers). Simi-

larly, **Figure 6.2** also shows the average lowest PSNR, highest PSNR and PSNR range achieved by each proposed method. The performance in terms of scalability in image quality degradation for every proposed method is discussed in the following paragraphs.

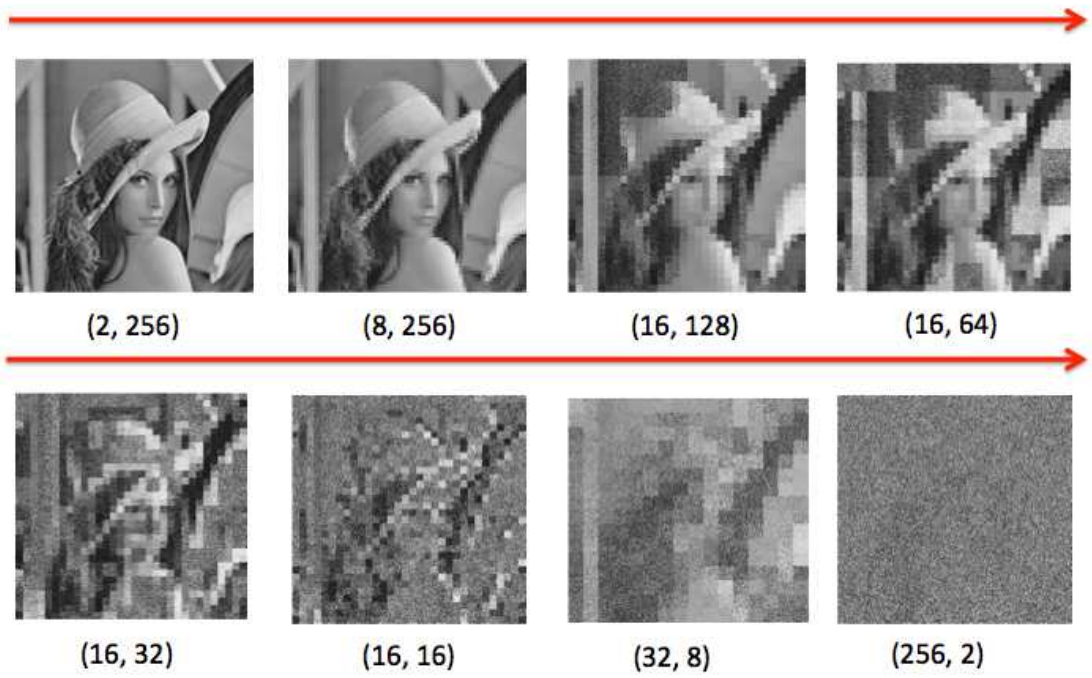


Figure 6.3: Combination of (b_s, b_ε) to achieve scalable image quality degradation

Scalability in HAM is achieved by controlling the embedding block size b_ε and the scrambling block size b_s. When b_ε or b_s increases, the distortion increases. Combinations of (b_s, b_ε) can produce a series of degraded images as shown in **Figure 6.3** by using the Lenna test image.

The highest achievable SSIM (PSNR) for HAM is 0.811 (25.83 dB) while the lowest achievable SSIM (PSNR) is 0.040 (11.54 dB). The achievable range of SSIM is the second widest (i.e., second most scalable) in comparison among all proposed methods, which is 0.771. Nevertheless, trial-and-errors are needed to fine-tune the two designated parameters to achieve the desired image distortion. Furthermore, some of the output images suffer from blockiness due to the block-based operation in HAM.

Scalability in SURIH is achieved by permuting the DC coefficients within windows

of predefined sizes. The achievable SSIM (PSNR) for SURIH ranges from 0.006 (6.22 dB) to 0.495 (21.63 dB). SURIH exhibits the smallest range of SSIM among all four methods, which is 0.489 only. In addition, SURIH has flexibility in producing high quality image because the highest achievable SSIM is 0.495 (21.63 dB). However, SURIH can produce the most distorted image among all the proposed method with SSIM value of 0.006 and PSNR value of 6.22 dB. Although SURIH can embed large of amount information (i.e., with high computational power needed), but it is less scalable, i.e., less flexible, than other methods in terms of image quality degradation.

Next, NIPS method is proposed to eliminate the blockiness and edginess problems in HAM. The scalability in NIPS can be easily controlled by restricting the number of pixels in a group P and number of times a row / column can be rotated $\omega_{\max}$. Example of scalable output for NIPS methods are shown in **Figure 5.3** and **Figure 5.4**. The range of PSNR appeared to be the smallest among all the four proposed methods. This is because the results are obtained using the combination of row and column operations. Although the embedding capacity for NIPS (itself) is not as high as HAM, NIPS is able to achieve smooth image quality degradation as compared to HAM while eliminating the blockiness and visible outline in HAM.

Lastly, the NIPC method is the most scalable method among the proposed methods because it has the largest range in both SSIM (i.e., 0.851) and PSNR (26.42 dB). The lowest achievable SSIM (PSNR) is 0.064 (10.05 dB) while the highest achievable SSIM (PSNR) is 0.915 (36.47 dB). The scalability in NIPC can be achieved by combining the proposed techniques and controlling the parameters. Although this is the most scalable method, it involves combinations of techniques and parameters, which makes it more difficult to fine-tune and seek for the desired distortion level. This is a drawback of being a highly flexible method.

In **Figure 6.1**, all the lowest achievable SSIM are near to 0, which also means that the

output images are severely distorted by using the proposed methods. Nevertheless, all the proposed methods are not able to approach unity (i.e., 1). Note that, all the information hiding algorithms embed data or scramble image by modifying the components (e.g., pixels, coefficients, etc.), which then lead to distortion. Since SSIM is very sensitive to the perceived changes of structural information in the image. Therefore, even the modifications are insignificant, reduction in SSIM is inevitable. To achieve high SSIM value (i.e., near to unity), structure of the image needs to be carefully studied to achieve a good trade-off between distortion and modification when designing a unified information hiding method.

6.2.2 Scalability in Embedding Capacity

All the proposed methods achieve scalability in embedding capacity. For HAM, the effective embedding capacity for Lenna image is in the range of 0.31 bpp ($b_\varepsilon = 512$) to 2.13 bpp ($b_\varepsilon = 16$). The same trend is also observed for a large dataset (i.e., Caltech 101 dataset) and the results are shown in **Table 3.6**.

Unlike other proposed methods, SURIH achieves scalability in embedding capacity by using a data representation scheme called VQD. Theoretically, SURIH is able to embed large amount of external data but it is restricted by the complexity of the VQD. As the representative example, the Lenna image is able to achieve gross embedding capacity ranges from 8649 bits ($n = 2$) to 18807 bits ($n = 6$). Nevertheless, n can be increased to accommodate more external data at the expense of higher complexity.

NIPS and NIPC achieve scalability in embedding capacity by following the S2E approach. NIPS itself can achieve 5812 bits (P = 1) to 8478 bits (P = 8) in the Lenna image. However, HAM can be utilized to increase the embedding capacity of NIPS. Meanwhile, NIPC can embed from 1131 to 15478 bits into the Lenna image. NIPS and NIPC method utilizes scrambling technique to embed data. However, the embedding

capacity in these two proposed methods are not as high as other approaches (i.e., E2S and STE). NIPS and NIPC are still able to achieve scalability in embedding capacity.

Overall, HAM method is able to achieve the highest embedding capacity among all the proposed methods. HAM exploits the maximum achievable embedding capacity by histogram manipulation method, that can be inserted into the image if there is no constraint on distortion. Besides, HAM is implemented in spatial domain and there is no concern on the bitstream size increment. Unlike HAM, NIPS and NIPC utilize scrambling algorithm to embed data. Therefore, the amount of data that can be embedded is restricted by the number of states (i.e., patterns) generated by using the proposed scrambling algorithm. Moreover, NIPC needs to be format-compliant and suppress bitstream size increment. Hence, its embedding capacity is lower than that of HAM method.

Scalability in image quality degradation and scalability in embedding capacity are inter-related. When image quality is preserved (i.e., low distortion), the embedding capacity is low. When the high image quality requirement is relaxed, higher capacity can be achieved.

6.3 Reversibility

If the application in questions relaxes the requirement on reversibility, more embedding capacity can be achieved. NIPC offers the flexibility to the user to choose to operate in reversible or rewritable mode. However, reversibility is one of the important requirements for most crucial image-based applications such as design draft, medical images, historical artworks, military images, etc. Therefore, all the four proposed methods in this study achieve reversible functionality by design.

6.4 Commutative

In general, commutative can be achieved in two simple ways, one is by separating the embedding and scrambling components, the other is by embedding external data into the

component which does not cause significant distortion (i.e., LSB). Commutative is important for UIH because it offers the flexibility to the encoder and decoder in performing the encoding (i.e., embedding first or scrambling first) and decoding (i.e., data extracting first or de-scrambling first) operations.

The proposed methods in spatial domain (i.e., HAM and NIPS) are not commutative (i.e., separable). This is because they are proposed to achieve scrambling-embedding in one single technique (i.e., either data embedding or scrambling technique) and they are operate in the entire image. Therefore, encoding and decoding process are done in a single step.

The proposed method following the STE approach (i.e., SURIH) in AC coefficients is not commutative because it has to be decoded using the reverse order of the encoding process.

The proposed NIPC techniques operates on three different entitles of JPEG compressed image. Each of them can be decoded by themselves without affecting other scrambling-embedding outcome in other areas. Therefore, the proposed NIPC method is separable in this way.

6.5 Limitation of this Study

In the ideal case, when scalability in scrambling and embedding capacity are tuned to low (i.e., lower distortion and lower embedding capacity), the proposed method is utilized as external data insertion method which can preserve the image quality. On the other hand, when both scalabilities are tuned to the highest level (i.e., highest distortion and highest embedding capacity), the proposed method is utilized as scrambling method which can embed large amount of external data. However, when our proposed method is tuned to low scalability, the method can be utilized for external data insertion purposes but the robustness against unauthorized viewing decreases and the embedding capacity is

relatively low.

Besides, all of the proposed methods involved few parameters. These parameters are utilized to adjust the desired image quality based on the intended application. In our current implementations, the possible image quality (i.e., from low distortion to high distortion) are investigated and this process is done by exhaustively by trying all the possible values. The exhaustive process is time and processing power consuming.

6.6 Summary

The important properties in UIH are discussed and compared for each proposed method. HAM is able to achieve high embedding capacity with lower complexity than SURIH, but its output is not smooth (i.e., blockiness and edginess). SURIH is able to embed theoretically the highest amount of external data using its data representation, but the computational cost is high. Meanwhile, NIPS eliminates the artifacts produced by HAM and it is able to embed high embedding capacity by integrating with HAM. Last but not least, NIPC is separable, reversible, and the most scalable method in terms of image quality degradation. Nevertheless, there are many parameters to fine-tune in order to produce the desired image quality.

CHAPTER 7

CONCLUSION

7.1 Summary of the Proposed Methods

This thesis studied the related literatures in two major disciplines of information hiding, namely, external data insertion methods and scrambling methods. A general framework was proposed to capture four different approaches to realize unified information hiding. Different approaches are pursued to propose four unified information hiding methods in the spatial and JPEG compressed domains, while solving the problems suffered by the conventional methods.

HAM

Histogram Association Mapping (HAM) is realized by using the proposed Embed-To-Scramble (E2S) approach. HAM embeds data into the image while purposely introducing distortion to the image. Using the proposed method, scalability in image quality and embedding capacity are achieved. In addition to achieve unification of both disciplines of information hiding, proposed HAM method is able to eliminate the expensive pre-processing problem and mitigate the overflow-underflow problem in the conventional histogram shifting methods. When all the bins in the histogram are fully occupied, HAM is still able to perform data embedding because it operates on each non-overlapping block of the image independently. Experimental results show that HAM is able to embed up to ~ 2.88 bpp (i.e., effective embedding capacity) using the CalTech 101 dataset. This observation also suggests that when the image quality requirement is relaxed (i.e., fully distorted), the maximum embeddable capacity is ~ 5.01 bpp (i.e., gross embedding capacity). As conclusion, HAM is able to achieve high embedding capacity, eliminate ex-

pensive pre-processing, overcome the overflow-underflow problem, achieve reversibility, and achieve scalability in image quality degradation and embedding capacity. Although the general and specific objectives are achieved, the output image suffers from blockiness and it fails to completely hide the visible outline of some images. Therefore, **Chapter 5** proposed another unified method to achieve scrambling-embedding in the spatial domain while producing smooth degradation.

SURIH

Scalable Unified Reversible Information Hiding (SURIH) is proposed following the Scramble-Then-Embed (STE) approach. SURIH aims to solve the bitstream size increment problem in the conventional information hiding methods by introducing adaptive scanning order. In addition, the scope of permutation is significantly increased to enhance its robustness as compared to the conventional scrambling methods. AC coefficient is first scrambled, then external data is embedded using the proposed Virtual Queue Decomposition (VQD) data representation scheme. Scalability in image quality degradation is then achieved by scrambling the DC coefficients. SURIH is reversible, scalable, and the scope of the permutation is significantly increased. Yet, the bitstream size increment in SURIH is $\sim 8.4\%$ by using the standard test images. This is because foreign ZRV pairs are introduced to the image to enable reversibility. Therefore, **Chapter 5** aimed to further suppress the bitstream size increment while achieving scrambling-embedding.

NIPS and NIPC

UIH using Natural Image Property in Spatial domain (NIPS) and UIH using Natural Image Properties in Compressed domain (NIPC) are proposed in **Chapter 5** to realize scrambling-embedding following the Scramble-To-Embed (S2E) approach. The challenge in the S2E approach is to exploit the natural image properties to materialize image recovery and extraction of the inserted data. For these purposes, NIPS utilized the correlation between pixels in the decoding process. NIPS is able to produce smooth output with

even distortion while achieving scalability in image quality degradation and embedding capacity. In NIPC, DC coefficient, AC block energy and ZRV pairs are utilized. The DC coefficient and AC block energy properties are similar to that of NIPS. Thus, the improved technique of NIPS is utilized in these two components to achieve scrambling-embedding. The properties of ZRV pairs are also investigated and derived for scrambling-embedding purposes. The output images provided by NIPC resemble noise, but the bitstream size increment is smaller than SURIH ($\sim 3.19\%$ using the standard test images). At the same time, scalability in image quality degradation, scalability in embedding capacity and reversibility are also achieved. NIPC is also robust against four considered sketch attacks.

In summary, this thesis proposed and designed four UIH methods following different approaches to achieve scrambling-embedding. Each of the methods has its own strengths and weaknesses. HAM is simple and high in embedding capacity, but its progression in image quality degradation is not smooth (i.e., visible blockiness). SURIH had broaden the scope of permutation to increase the robustness level and achieves theoretically the highest embedding capacity, but the computational cost is expensive and bitstream size increment is inevitable due to the introduction of foreign ZRV pairs. Although NIPS is able to provide smoother degradation than HAM, the embedding capacity is relatively lower than HAM. For NIPC, it is commutative and able to further suppress bitstream increment caused by scrambling-embedding, but the embedding capacity is also relatively lower than SURIH. Therefore, these proposed reversible UIH methods should be selectively utilized depending on the requirements of the application in question. Here, only the fundamental and analysis of each approaches are discussed, but each method still needs to be fine-tune to serve the specific requirements of the application in question.

7.2 Contributions

In this thesis, we proposed a unified information hiding frameworks for any information hiding methods. Based on the proposed UIH framework, four possible approaches are derived to realize unified information hiding, including Embed-Then-Scramble (ETS), Scramble-Then-Embed (STE), Embed-To-Scramble (E2S), and Scramble-To-Embed (S2E). Among them, E2S and S2E are the novel approaches to achieve scrambling-embedding. In addition, important properties such as scalability, reversibility and commutative in UIH are also discussed.

Other than that, problems of current information hiding techniques (i.e., scrambling and data insertion techniques) are also addressed and solved by using the proposed methods following different approaches to achieve scrambling-embedding.

HAM

A novel histogram association method (HAM) is proposed by following E2S approach. HAM is operated in block-based manner and able to perform one-to-many mapping. It is also able solve the problems in previous histogram manipulation method (i.e., as mentioned in **Chapter 3**) to:

- eliminate pre-processing (i.e., shifting)
- achieve higher embedding capacity
- eliminate overflow-underflow, and
- embeddable even when the histogram bins are fully occupied.

SURIH

Adaptive scanning order and virtual queue decomposition (VQD) are proposed in SURIH following STE approach to realize scrambling-embedding. Besides, it is also

able to solve the problems in information hiding methods in frequency domain (i.e., as mentioned in **Chapter 4**) to:

- increase scope of permutation, and
- suppress bitstream size increment.

NIPS and NIPC

Natural image properties in both spatial and frequency domains are studied to realize scrambling-embedding following S2E approach. It is able to solve the problems identified in **Chapter 5** to:

- achieve smooth image degradation,
- increase scope of permutation further,
- suppress bitstream size further, and
- utilize all components for scrambling-embedding.

7.3 Advantages and Disadvantages of each Proposed Methods

All the proposed methods are reversible, scalable and able to achieve unification in information hiding (i.e., scrambling-embedding). **Table 7.1** summarizes the advantages and disadvantages of each proposed methods.

7.4 Future Works

There are still rooms for further improvement and development in the unified information hiding approaches, particularly in the E2S and S2E approaches. To the best of our knowledge, our proposed methods that pursue these two approaches are the novel methods introduced to the UIH research area. While the concepts are realized and verified in this study by using still image, these proposed methods can be extended to the compressed video domain, such as H.264/AVC and High Efficiency Video Coding (HEVC).

[t]

Table 7.1: Advantages and Disadvantages of each methods

Method	Advantages	Disadvantages
HAM (**Chapter 3**)	Eliminate expensive pre-processing Eliminate overflow-underflow problem Achieve high embedding capacity	Resemble blockiness artifacts
SURIH (**Chapter 4**)	Increase scope of permutation Reduce bitstream size increment Further embed without increasing bitstream size Survive in sketch attacks	Bitstream size increment Some components are not utilized
NIPS and NIPC (**Chapter 5**)	Further increase scope of permutation Produce smooth quality degradation Further suppress bitstream size Robust against sketch attacks	Limited embedding capacity

In addition, the natural properties in the motion vectors and video frames can be further explored to design an UIH method for achieving scrambling and data insertion simultaneously. Last but not least, without relying on manual tuning of the parameter values, adaptive UIH methods will be devised to meet user's desired visual quality and embedding capacity.

Appendices

APPENDIX A

PROOF FOR EQUATION (4.9) IN CHAPTER 4

Here, we prove **Equation (4.9)**. Recall that the number of theoretically possible decompositions of t items (i.e., zrv pairs) into n sub-queues is denoted by $\tau(n,t)$. We start with n sub-queues with $t-1$ items, then add one more item to find the value $\tau(n,t)$. We consider two ways of inserting the new item, namely, (i.) at the last sub-queue, and (ii.) at any sub-queue other than the last one. If we put the newly added item at the last sub-queue, we have exactly $\tau(n,t-1)$ decompositions. By doing so, the last sub-queue will always have at least one item in it. In other words, we have considered all theoretically possible decompositions of t items into n sub-queues where the last sub-queue is non-empty. Next, we consider the number of decompositions of t items into n sub-queues where the last sub-queue is always empty. This value is exactly the same as $\tau(n-1,t)$. Therefore, $\tau(n,t) = \tau(n-1,t) + \tau(n,t-1)$ holds true.

APPENDIX B

SKETCH ATTACKS

DC Category Mapping Attack (DCM) (Ong et al., 2013) is performed by assigning a representative values (e.g., 2^x for category x) to each category of the DPCM prediction error value. DCM is effective for method which maps the (DC or AC) coefficient values to another values from the same category in the JPEG Huffman Table.

Minemura et al. also proposed three simple sketch attacks to estimate the outline information from the AC coefficients (Minemura et al., 2012). These simple sketch attack includes Nonzero Coefficient Attack (NCC), Energy of AC Coefficients (EAC) and Position of Last Nonzero AC Coefficient (PLZ), and they are detailed as follows to generate edge image G^a of size $M \times N$:

NCC: Nonzero Coefficient Attack

$$G^a_{NCC}(i,j) \leftarrow round\left\{255 \times \frac{\zeta(i,j)}{\max_{(i,j)}\{\zeta(i,j)\}}\right\}, \tag{B.1}$$

where $\zeta(i,j)$ is the number of nonzero coefficients in the (i,j)-th block (refer to **Section 4.2.1**).

EAC: Energy of AC Coefficients

$$G^a_{EAC}(i,j) \leftarrow \frac{\Lambda(i,j)}{\sum_{i=1}^{M}\sum_{j=1}^{N}[\Lambda(i,j)/(M \times N)]}, \tag{B.2}$$

where $\Lambda(i,j)$ is the sum of magnitude of AC coefficients in the (i,j)-th block as indicated by **Equation (5.10)**.

PLZ: Position of Last Nonzero AC Coefficient

$$G^{a}_{PLZ}(i,j) \leftarrow \frac{G_{u_0,v_0}(i,j)}{\max_{(i,j)}\{G_{u_0,v_0}(i,j)\}}, \tag{B.3}$$

where $G_{u_0,v_0}(i,j)$ is the position of the last nonzero AC coefficients in the (i,j)-th block using standard zigzag order.

APPENDIX C

ABBREVIATIONS

Table C.1: List of Abbreviations

Abbreviation	Description
AC	Alternating Current (coefficient)
bpnc	bits per nonzero coefficient
BpNAC	Bits per Nonzero AC coefficient
BpDC	Bits per DC coefficient
dB	decimal Base
DC	Direct Current (coefficient)
DCT	Discrete Cosine Transform
E2S	Embed-To-Scramble
ETS	Embed-Then-Scramble
FIBS	Full Inter-Block Shuffling
HAM	Histogram Association Mapping
HVS	Human Visual Systems
JPEG	Joint Picture Expert Group
LSB	Least Significant Bit
NRB	Non-Reflective Block
NRB T1	NRB Enhancement Method 1
NRB T2	NRB Enhancement Method 2
NZCA	Non Zero Coefficient count Attack
PSNR	Pixel Signal-to-Noise Ratio
QF	Quality Factor
QT	Quantization Table
RB	Reflective Block
RRBE	Reserving Room Before Embedding
S2E	Scramble-To-Embed
STE	Scramble-Then-Embed
SSIM	Structural Similarity Index Measurement
SURIH	Scalable Unified Reversible Information Hiding
UCPF-D	Unified Constructive Permutation Function (Diagonal)
UCPF-VH	Unified Constructive Permutation Function (Vertical, Horizontal)
UIH	Unified Information Hiding
VQD	Virtual Queue Decomposition
w.r.t.	with respect to
ZRV	Zero-Run Value pairs

APPENDIX D

PUBLICATIONS

Following are the list of submitted and accepted journal papers and conference papers related to the works done in this thesis.

D.1 Journals

i S. Ong, K. Wong, and K. Tanaka, "Scrambling-Embedding for JPEG Compressed Image," Signal Processing, 109 (0), pp. 38-53, April 2015.

ii S. Ong, K. Wong, X. Qi, and K. Tanaka, "Beyond Perceptual Encryption for JPEG Image," Signal Processing:Image Communication, 31 (0), pp. 47-60, February 2015.

iii S. Ong, K. Wong, and K. Tanaka, "A Scalable Reversible Data Embedding Method with Progressive Quality Degradation Functionality," Signal Processing: Image Communication, 29 (1), pp. 135-149, January 2014.

D.2 Conferences and Proceedings

i S. Ong, K. Wong, and K. Tanaka, "Reversible and Tunable Scrambling-Embedding Method," Intelligent Signal Processing and Communication Systems (ISPACS), pp. 608 - 613, Naha, Japan, 12 - 15 November 2013. (Accepted for oral presentation).

ii S. Ong, and K. Wong, "Rotational based Rewritable Data Hiding in JPEG," IEEE Visual Communications and Image Processing (VCIP), pp. 1-6, Kuching, Malaysia, 17 - 20 November 2013. (Submitted and accepted for poster presentation)

iii S. Ong, K. Minemura, and K. Wong, "Progressive Quality Degradation in JPEG Compressed Image using DC Block Orientation with Rewritable Data Embedding

Functionality", International Conference on Image Processing (ICIP), pp. 4574-4578, Melbourne, Australia, 15 - 18 September 2013. (Accepted for poster presentation)

iv S. Ong, K. Wong, and K. Tanaka, "Improvement on Reversible Data Embedding Method using Virtual Queue Decomposition", IIEEJ 3rd Image Electronics and Image Computing Workshop (IEVC), Kuching, Malaysia, 21-24 November 2012. (Accepted for oral presentation, Excellent Paper Award)

v S. Ong, K. Wong, and K. Tanaka, "Reversible Data Embedding using Reflective Blocks with Scalable Visual Quality Degradation", 2012 IEEE International Conference on International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Piraeus, Greece, 18-20 July 2012. (Accepted for oral presentation)

vi K. Wong, S. Ong, and K. Tanaka, "Improvement of carrier capacity for scalable scrambling method with reversible information insertion functionality", 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 312 317, Kuala Lumpur, Malaysia, 16 - 18 November 2011. (Accepted for oral presentation)

vii K. Wong; S. Ong; K. Tanaka and X. Qi; ,"Bitstream Size Suppression for DCT-based Information Hiding Method", 2011 IEEE Conference on International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), ChiangMai, Thailand, 2011. (Accepted for oral presentation, Best Paper Award - Signal Processing Track)

REFERENCES

197, F. I. P. S. P. (2001, November). *Announcing the advanced encryption standard (AES)* (Tech. Rep.). United States National Institute of Standards and Technology (NIST).

46-3, F. I. P. S. P. (1999, October). *Data encryption standard (DES)* (Tech. Rep.). United States National Institute of Standards and Technology (NIST).

Battisti, F., Cancellaro, M., Boato, G., Carli, M., & Neri, A. (2009, January). Joint watermarking and encryption of color images in the Fibonacci-Haar domain. *EURASIP Journal on Advances in Signal Processing*, *2009*, 43:3 - 43:3. doi: 10.1155/2009/938515

Bhargava, B., Shi, C., & Wang, S.-Y. (2004). MPEG video encryption algorithms. *Multimedia Tools and Applications*, *24*(1), 57 - 79. doi: 10.1023/B:MTAP.0000033983.62130.00

Bodo, Y., Laurent, N., & Dugelay, J.-L. (2004, September). Watermarking video, hierachical embedding in motion vectors. In *IEEE international conference on image processing* (p. 739 - 742).

Bouslimi, D., Coatrieux, G., & Roux, C. (2011). A joint watermarking/encryption algorithm for verifying medical image integrity and authenticity in both encrypted and spatial domains. In *Ieee annual international conference of the engineering in medicine and biology society* (p. 8066 - 8069).

Celik, M. U., Sharma, G., Tekalp, A. M., & Saber, E. (2005). Lossless generalized-LSB data embedding. *IEEE Transactions on Image Processing*, *14*(2), 253 - 266.

Chan, C.-K., & Cheng, L. (2004). Hiding data in images by simple LSB substitution. *Pattern Recognition*, *37*(3), 469 - 474.

Cheng, H., & Li, X. (2000, August). Partial encryption of compressed images and videos. *IEEE Transactions on Signal Processing*, *48*(8), 2439 - 2451.

Coltuc, D. (2011). Improved embedding for prediction-based reversible watermarking. *IEEE Transactions on Information Forensics and Security*, *6*(3 - 2), 873 - 882.

De Vleeschouwer, C., Delaigle, J. F., & Macq, B. (2003, March). Circular interpretation of bijective transformations in lossless watermarking for media asset management. *IEEE Transactions on Multimedia*, *5*(1), 97 - 105.

Dragoi, I.-C., & Coltuc, D. (2014, April). Local-prediction-based difference expansion reversible watermarking. *IEEE Transactions on Image Processing*, *23*(4), 1779 - 1790.

Dufaux, F., & Ebrahimi, T. (2008, August). Scrambling for privacy protection in video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(8), 1168 - 1174.

Engel, D., Pschernig, E., & Uhl, A. (2008). An analysis of lightweight encryption schemes for fingerprint images. *IEEE Transactions on Information Forensics and Security*, *3*(2), 173 - 182.

Fei-Fei, L., Fergus, R., & Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, *106*(1), 59 - 70. (Special issue on Generative Model Based Vision) doi: 10.1016/j.cviu.2005.09.012

Fridrich, J., Goljan, M., & Du, R. (1900). Lossless data embedding - new paradigm in digital watermarking. *EURASIP Journal on Advances in Signal Processing*, *2002*(2), 185 - 196.

Fridrich, J., Goljan, M., & Du, R. (2001). Invertible authentication watermark for JPEG images. In *International conference on information technology: Coding and computing* (p. 223 - 227).

Fujiyoshi, M. (2013, July). Separable reversible data hiding in encrypted images with histogram per-

mutation. In *Ieee international conference on multimedia and expo workshops* (p. 1 - 4). doi: 10.1109/ICMEW.2013.6618266

Fujiyoshi, M., Kuroiwa, K., & Kiya, H. (2008). A scrambling method for motion JPEG videos enabling moving objects detection from scrambled videos. In *Ieee international conference on image processing* (p. 773 - 776).

Goljan, M., Fridrich, J., & Du, R. (2001). Distortion-free data embedding for images. In I. Moskowitz (Ed.), *Information hiding* (Vol. 2137, p. 27 - 41). Springer Berlin Heidelberg.

Golomb, S. (1966, July). Run-length encodings. *IEEE Transactions on Information Theory*, *12*(3), 399 - 401.

Grangetto, M., Magli, E., & Olmo, G. (2006). Multimedia selective encryption by means of randomized arithmetic coding. *IEEE Transactions on Multimedia*, *8*(5), 905 - 917.

Hong, W. (2012). Human visual system based data embedding method using quadtree partitioning. *Signal Processing: Image Communication*, *27*(10), 1123 - 1133.

Hu, J., & Han, F. (2009). A pixel-based scrambling scheme for digital medical images protection. *Journal of Network and Computer Applications*, *32*(4), 788 - 794.

Huang, H.-C., & Fang, W.-C. (2011). Authenticity preservation with histogram-based reversible data hiding and quadtree concepts. *Sensors*, *11*(10), 9717 - 9731.

IACP, center for social media. (2014). Retrieved 28 April 2014, from `www.iacpsocialmedia.org/Resources/FunFacts.aspx`.

Itoh, Y. (2003, June). An adaptive DCT coding with geometrical edge representation. *IEICE Transactions on Information and Systems*, *E86-D*(6), 1087 - 1094.

Jose, R., & Abraham, G. (2013, June). A separable reversible data hiding in encrypted image with improved performance. In *Annual international conference on emerging research areas and 2013 international conference on microelectronics, communications and renewable energy (aicera/icmicr)* (p. 1 - 5).

Jung, S.-W., Ha, L. T., & Ko, S.-J. (2011). A new histogram modification based reversible data hiding algorithm considering the human visual system. *IEEE Signal Processing Letters*, *18*(2), 95 - 98.

Kailasanathan, C., & Naini, R. (2002). Compression performance of JPEG encryption scheme. In *International conference on digital signal processing* (Vol. 2, p. 1329 - 1332). doi: 10.1109/ICDSP.2002.1028339

Kankanhalli, M. S., & Guan, T. T. (2002, May). Compressed-domain scrambler/descrambler for digital video. *IEEE Transactions on Consumer Electronics*, *48*(2), 356 - 365.

Karim, M. S. A., & Wong, K. (2014). Universal data embedding in encrypted domain. *Signal Processing*, *94*(0), 174 - 182.

Katzenbeisser, S., & Petitcolas, F. (2000). *Information hiding techniques for steganography and digital watermarking*. Artech House Publishers.

Kawaguchi, E., & Eason, R. O. (1998, November). Principle and applications of BPCS-steganography. In *Proceedings of spie: Multimedia systems and applications* (Vol. 3528, p. 464 - 473). Boston.

Kundur, D., & Karthik, K. (2004). Video fingerprinting and encryption principles for digital rights management. *Proceedings of the IEEE*, *92*(6), 918 - 932.

Lee, C.-F., & Chen, H.-L. (2012). Adjustable prediction-based reversible data hiding. *Digital Signal Processing*, *22*(6), 941 - 953.

Lee, C.-W., & Tsai, W.-H. (2013). A data hiding method based on information sharing via PNG images for

applications of color image authentication and metadata embedding. *Signal Processing*, *93*(7), 2010 - 2025.

Li, S., Chen, G., Cheung, A., Bhargava, B., & Lo, K.-T. (2007, February). On the design of perceptual MPEG-video encryption algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*(2), 214 - 223.

Li, W., & Yuan, Y. (2007, September). A leak and its remedy in JPEG image encryption. *International Journal of Computer Mathematics - Computer Vision and Pattern Recognition*, *84*, 1367 - 1378.

Li, X., Li, B., Yang, B., & Zeng, T. (2013, June). General framework to histogram-shifting-based reversible data hiding. *IEEE Transactions on Image Processing*, *22*(6), 2181 - 2191.

Lian, S., Liu, Z., Ren, Z., & Wang, H. (2007). Commutative encryption and watermarking in video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*, 774 - 778.

Lin, C.-C., Chen, S.-C., & Hsueh, N.-L. (2009, January). Adaptive embedding techniques for VQ-compressed images. *Information Sciences*, *179*, 140 - 149.

Lin, C.-Y., Chang, C.-C., & Wang, Y.-Z. (2008, March). Reversible steganographic method with high payload for JPEG images. *IEICE Transactions on Information and Systems*, *E91-D*, 836 - 845.

Liu, X., & Eskicioglu, A. M. (2003). Selective encryption of multimedia content in distribution networks: challenges and new directions. In *International conference on communications, internet, and information technology* (p. 527 - 533).

Liu, Z., Liang, H., Niu, X., & Yang, Y. (2004, September). A robust video watermarking in motion vectors. In *IEEE international conference on signal processing* (Vol. 3, p. 2358 - 2361).

Luo, W., Huang, F., & Huang, J. (2010). Edge adaptive image steganography based on LSB matching revisited. *IEEE Transactions on Information Forensics and Security*, *5*(2), 201 - 214.

Lusson, F., Bailey, K., Leeney, M., & Curran, K. (2013). A novel approach to digital watermarking, exploiting colour spaces. *Signal Processing*, *93*(5), 1268 - 1294.

Ma, K., Zhang, W., Zhao, X., Yu, N., & Li, F. (2013). Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Transactions on Information Forensics and Security*, *8*(3), 553 - 562.

Massoudi, A., Lefebvre, F., De Vleeschouwer, C., Macq, B., & Quisquater, J.-J. (2008, January). Overview on selective encryption of image and video: Challenges and perspectives. *EURASIP Journal on Information Security*, *2008*, 5:1 - 5:18. Retrieved from `http://dx.doi.org/10.1155/2008/179290` doi: 10.1155/2008/179290

Memon, N. D., & Wong, P. W. (2001). A buyer-seller watermarking protocol. *IEEE Transactions on Image Processing*, *10*(4), 643 - 649.

Minemura, K., Moayed, Z., Wong, K., Qi, X., & Tanaka, K. (2012). JPEG image scrambling without expansion in bitstream size. In *Ieee international conference on image processing* (p. 261 - 264).

Mobasseri, B., Berger, R., Marcinak, M., & NaikRaikar, Y. (2010). Data embedding in JPEG bitstream by code mapping. *IEEE Transactions on Image Processing*, *19*(4), 958 - 966. doi: 10.1109/TIP.2009.2035227

Neil F. Johnson, Z. D., & Jajodia, S. (2001). *Information hiding steganography and watermarking attacks and countermeasures*. Norwell, MA, USA: Kluwer Academic Publishers.

Ni, Z., Shi, Y.-Q., Ansari, N., & Su, W. (2006). Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, *16*(3), 354 - 362.

Niu, X., Zhou, C., Ding, J., & Yang, B. (2008, August). JPEG encryption with file size preservation. In

International conference on intelligent information hiding and multimedia signal processing (p. 308 - 311).

Ogihara, T., Nakamura, D., & Yokoya, N. (1996, August). Data embedding into pictorial images with less distortion using discrete cosine transform. In *Proceedings of the 13th international conference on pattern recognition* (Vol. 2, p. 675 - 679).

Ong, S., Minemura, K., & Wong, K. (2013, September). Progressive quality degradation in JPEG compressed image using DC block orientation with rewritable data embedding functionality. In *Ieee international conference on image processing* (p. 4574 - 4578).

Ong, S., Wong, K., & Tanaka, K. (2012, July). Reversible data embedding using reflective blocks with scalable visual quality degradation. In *International conference on intelligent information hiding and multimedia signal processing* (p. 363 - 366).

Pan, F. (2002, December). Adaptive image compression using local pattern information. *Pattern Recognition Letter*, *23*, 1837 - 1845.

Pazarci, M., & Dipcin, V. (2002, May). A MPEG2-transparent scrambling technique. *IEEE Transactions on Consumer Electronics*, *48*(2), 345 - 355.

Pennebaker, W. B., & Mitchell, J. L. (1992). *JPEG still image data compression standard*. New York: Van Nostrand Reinhold.

Qian, Z., Zhang, X., & Wang, S. (2014). Reversible data hiding in encrypted jpeg bitstream. *IEEE Transactions on Multimedia*.

Rad, R., Wong, K., & Guo, J.-M. (2014, April). A unified data embedding and scrambling method. *IEEE Transactions on Image Processing*, *23*(4), 1463 - 1475.

Schaefer, G., & Stich, M. (2004). UCID - an uncompressed colour image database. In *In storage and retrieval methods and applications for multimedia 2004, proceedings of spie* (Vol. 5307, p. 472 - 480).

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell system technical journal*, *28*(4), 656 - 715.

Shi, C., & Bhargava, B. (1998). A fast MPEG video encryption algorithm. In *Proceedings of the sixth acm international conference on multimedia* (p. 81 - 88).

Sur, A., Goel, P., & Mukhopadhyay, J. (2008, March). A spatial domain steganography scheme for reducing embedding noise. In *International symposium on communications, control and signal processing* (p. 1029 - 1034).

T.81 : Information technology - digital compression and coding of continuous-tone still images - requirements and guidelines [online]. (1992). Retrieved from `http://www.itu.int/rec/T-REC-T.81/en`

Takayama, M., Tanaka, K., Nakajima, Y., & Kouchi, T. (2008, March). A scalable video scrambling method in MPEG compressed domain. In *International symposium on communications, control and signal processing* (p. 1035 - 1040).

Takayama, M., Tanaka, K., Yoneyama, A., & Nakajima, Y. (2006, July). A video scrambling scheme applicable to local region without data expansion. In *Ieee international conference on multimedia and expo* (p. 1349 - 1352).

Tang, L. (1996). Methods for encrypting and decrypting MPEG video data efficiently. In *Proceedings of the fourth acm international conference on multimedia* (p. 219 - 229). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/244130.244209` doi: 10.1145/244130.244209

Tew, Y., & Wong, K. (2014, Feb). An overview of information hiding in H.264/AVC compressed video.

IEEE Transactions on Circuits and Systems for Video Technology, *24*(2), 305 - 319.

Thai, T., Cogranne, R., & Retraint, F. (2014, May). Statistical model of quantized dct coefficients: Application in the steganalysis of jsteg algorithm. *IEEE Transactions on Image Processing,*, *23*(5), 1980 - 1993. doi: 10.1109/TIP.2014.2310126

Thodi, D., & Rodriguez, J. (2004). Reversible watermarking by prediction-error expansion. In *Ieee southwest symposium on image analysis and interpretation* (p. 21 - 25).

Tian, J. (2002). Reversible watermarking by difference expansion. In *Proceedings of workshop on multimedia and security* (p. 19 - 22).

Tian, J. (2003, August). Reversible data embedding using a difference expansion. *IEEE Transactions on Circuits and Systems for Video Technology*, *13*(8), 890 - 896.

Tian, J., & Wells, J., R.O. (2004). Reversible data-embedding with a hierarchical structure. In *Ieee international conference on image processing* (Vol. 5, p. 3419 - 3422).

Upham, D. (1999). *Jsteg steganographic algorithm.*

The USC-SIPI image database [online]. (n.d.). Retrieved from `http://sipi.usc.edu/database/`

Van De Ville, D., Philips, W., Van De Walle, R., & Lemahieu, I. (2004). Image scrambling without bandwidth expansion. *IEEE Transactions on Circuits and Systems for Video Technology*, *14*(6), 892 - 897.

Wang, C., Yu, H., & Zheng, M. (2003). A DCT-based MPEG-2 transparent scrambling algorithm. *IEEE Transactions on Consumer Electronics*, *49*, 1208 - 1213.

Wang, Y., O'Neill, M., & Kurugollu, F. (2013, September). A tunable encryption scheme and analysis of fast selective encryption for cavlc and cabac in h.264/avc. *IEEE Transactions on Circuits and Systems for Video Technology*, *23*(9), 1476 - 1490.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004, April). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600 - 612.

Wayner, P. (2009). *Disappearing cryptography: information hiding: steganography & watermarking*. Morgan Kaufmann.

Wong, K., & Tanaka, K. (2010a, December). Scalable image scrambling method using unified constructive permutation function on diagonal blocks. In *Picture coding symposium (pcs), 2010* (p. 138 -141).

Wong, K., & Tanaka, K. (2010b). Scalable scrambling method using unified constructive permutation function. In *Proceedings of the iieej image electronics and visual computing workshop.*

Wu, X., & Sun, W. (2014). High-capacity reversible data hiding in encrypted images by prediction error. *Signal Processing*, *104*(0), 387 - 400.

Wu, Y., Zhou, Y., Noonan, J. P., Panetta, K., & Agaian, S. (2010). Image encryption using the sudoku matrix. In *Spie defense, security, and sensing* (p. 77080P-1 - 77080P-12).

Xu, D., Wang, R., & Shi, Y. (2014, April). Data hiding in encrypted h.264/avc video streams by codeword substitution. *IEEE Transactions on Information Forensics and Security*, *9*(4), 596 - 606.

Xuan, G., Shi, Y. Q., Chai, P., Teng, J., Ni, Z., & Tong, X. (2009). Optimum histogram pair based image lossless data embedding. *Transactions on Data Hiding and Multimedia Security IV*, *4*, 84 - 102.

Yang, M., Bourbakis, N., & Li, S. (2004, August). Data-image-video encryption. *IEEE Potentials*, *23*(3), 28 - 34. doi: 10.1109/MP.2004.1341784

Zeng, W., & Lei, S. (2003). Efficient frequency domain selective scrambling of digital video. *IEEE*

Transactions on Multimedia, *5*(1), 118 - 129.

Zeng, Y.-C., Huang, F., & Liao, H.-Y. (2011). Compression and protection of JPEG images. In *Ieee international conference on image processing* (p. 2733 - 2736).

Zhang, J., Li, J., & Zhang, L. (2001, October). Video watermark technique in motion vector. In *IEEE brazilian symposium on computer graphics and image processing* (p. 179 - 182).

Zhang, X. (2012). Separable reversible data hiding in encrypted image. *IEEE Transactions on Information Forensics and Security*, *7*(2), 826 - 832.

Zhang, X., & Wang, S. (2005). Steganography using multiple-base notational system and human vision sensitivity. *IEEE Signal Processing Letters*, *12*(1), 67 - 70.